**Error! Switch argument not specified.**

Deep Space Network

## Software Operator's Manual

# Subsystem Interface Verifier

## DSN Operational Internal

| Prepared by: | Approved by: |
|---|---|
| John Veregge Date | John Veregge<br>Software Cognizant Development Engineer |

| Released by: | Belinda M Wilkinson Date<br>Operability Engineer | DSN No. SOM-DOI-5249-OP Rev. A<br>For Software Ver. TP-B/1.3.1<br>Issue Date October 12, 1995<br>NASA Ident. No. 23835 |
|---|---|---|
| Angie Cain Date<br>DSN Software Document Control | Chaw-Kwei Hung Date<br>Task Manager | Jet Propulsion Laboratory<br>California Institute of Technology |

# Review Page

Reviewed by:

**Required Reviewers**

| | | | |
|---|---|---|---|
| Larry Babb | Date | N/A | Date |
| System Cognizant Operations Engineer | | Major Assy. Cognizant Development Engineer | |

**Additional Reviewers**

Not applicable.

## Change Incorporation Log

| TP | Rev | Status | Pages Affected | Issue Date | Section |
|---|---|---|---|---|---|
| B | A | Draft | All (189 total pages) | 10/12/95 | 394 |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

# DISTRIBUTION

| | | |
|---|---|---|
| DSN Document Center | 503-102 | (2 Reg) |
| *SPC10 | | (2 Reg and 3 Small) |
| *SPC40 | | (3 Reg and 4 Small) |
| *SPC60 | | (3 Reg and 2 Small) |
| DTF-21 | 125-B17 | (1 Reg) |
| Hewitt, A. | 507-120 | (1 Reg) |
| Jones, J. | 303-403 | (1 Reg) |
| JPL MIL-71 | 125-B17 | (1 Reg) |
| Vault | 111-B25 | (Original) |
| Veregge, J. | 525-3601 | (1 Reg) |
| Wilkinson, B. | 525-3680 | (1 Reg and 1 Small) |

*Distribution to be handled through DSN Document Center (DDC); Mail Stop 503-102.

# TABLE OF CONTENTS


| Section | Title | Page |
|---|---|---|
| 1 | OVERVIEW AND OPERATING MODES | 1-1 |
| 1.1 | Overview and Operating Modes | 1-1 |
| 1.2 | Overview | 1-1 |
| 1.3 | Document Scope | 1-2 |
| 1.4 | Equipment Setup | 1-2 |
| 1.5 | Conventions and Notations | 1-2 |
| 1.5.1 | Conventions | 1-2 |
| 1.5.1.1 | Operator Directives Conventions | 1-3 |
| 1.5.1.2 | Display Conventions | 1-3 |
| 1.5.1.3 | Message Conventions | 1-4 |
| 1.5.2 | Notation | 1-5 |
| 1.5.2.1 | Operator Directive Syntax Notation | 1-5 |
| 1.5.2.2 | Display Notation | 1-6 |
| 1.5.2.3 | Message Notation | 1-7 |
| 1.5.2.4 | Example Notation | 1-7 |
| 1.6 | Controlling Document | 1-8 |
| 1.7 | Applicable Documents | 1-8 |
| 2 | OPERATOR DIRECTIVES | 2-1 |
| 2.1 | Operator Directives - Quick Reference | 2-1 |
| 2.2 | Menus Hierarchy | 2-50 |
| 2.3 | Menus - Detailed Description | 2-50 |
| 2.4 | Icons | 2-50 |
| 2.5 | Forms | 2-50 |
| 2.6 | Function Keys | 2-50 |
| 2.7 | Other Interactions | 2-50 |

TABLE OF CONTENTS (Continued)


| Section | Title | Page |
|---|---|---|
| 3 | MONITORING AND STATUS DISPLAYS | 3-1 |
| 3.1 | Display - Quick Reference | 3-1 |
| 3.2 | Displays - Hierarchy | 3-2 |
| 3.3 | Displays - Detailed Descriptions | 3-4 |
| 4 | MESSAGES | 4-1 |
| 4.1.1 | Emergency Alarms | 4-1 |
| 4.1.2 | Critical Alarms | 4-1 |
| 4.1.3 | Warning Alarms | 4-1 |
| 4.2 | Advisories | 4-2 |
| 4.2.1 | Deviation Advisories | 4-2 |
| 4.2.2 | Completion Advisories | 4-5 |
| 4.2.3 | Progress Advisories | 4-5 |
| 4.2.4 | Log-Only Advisories | 4-5 |
| 4.3 | Prompts | 4-6 |
| 5 | REPORTS | 5-1 |
| 5.1 | Reports - Quick Reference | 5-1 |
| 5.2 | Reports - Detailed Descriptions | 5-2 |
| 6 | RAPID INTERFACE DEFINITION (RID) | 6-1 |
| 6.1 | RID ASCII File Format | 6-1 |
| 6.1.1 | Transmission Record Description | 6-4 |
| 6.1.1.1 | Transmission Record Keyword Descriptions | 6-6 |
| 6.1.2 | Field Record Description | 6-10 |
| 6.1.2.1 | Data Type Descriptions | 6-16 |
| 6.1.2.2 | Action Descriptions | 6-20 |
| 6.1.3 | Raw Data Record Description | 6-34 |

6.2 Mapping an Interface to a RID ........ 6-36

TABLE OF CONTENTS (Continued)

| Section | Title | Page |
|---|---|---|
| 6.3 | RID Generation Tools | 6-44 |
| 6.3.1 | Text Editors | 6-44 |
| 6.3.2 | RID Lint | 6-45 |
| 6.3.3 | RID Editor | 6-47 |
| 6.3.4 | Subsystem Interface Agreement Translator | 6-48 |
| 6.3.5 | Subsystem Interface Agreement Definition | 6-53 |
| 6.4 | Changes to the RID Definition | 6-53 |
| 6.5 | Backward Compatible Features | 6-59 |
| 7 | USER'S GUIDE | 7-1 |
| 7.1 | Configuration | 7-1 |
| 7.1.1 | SIV In Your Development Lab | 7-1 |
| 7.1.1.1 | SIV Software Installation | 7-1 |
| 7.1.1.2 | Configuring Data Files used by SIV | 7-14 |
| 7.1.1.3 | Activating SIV Software | 7-16 |
| 7.1.1.4 | Terminating SIV Software | 7-18 |
| 7.1.1.5 | Running Multiple SIV Machines | 7-18 |
| 7.1.2 | SIV at the JPL Development Test Facility - DTF-21 | 7-18 |
| 7.1.2.1 | What You Need at DTF-21 | 7-19 |
| 7.2 | SIV Operating Configurations | 7-19 |
| 7.2.1 | Traditional Configuration | 7-20 |
| 7.2.2 | Link-Oriented Configuration | 7-21 |
| 7.3 | Automated Interface Testing with Auto-Tester | 7-22 |
| 7.3.1 | Examples of Auto-Tester Use | 7-22 |
| 7.3.2 | Test Script Format | 7-23 |
| 7.3.3 | Test Logs | 7-25 |

APPENDIX A NEW FEATURES....................................................A-1

# LIST OF FIGURES


| Figure | Title | Page |
|---|---|---|
| 3-1 | Subsystem Configuration Display | 3-4 |
| 3-2 | ISB Dump Display | 3-6 |
| 3-3 | RID Interface Control Table Display | 3-8 |
| 3-4 | Stream Status Display | 3-10 |
| 3-5 | SIV Status Display | 3-12 |
| 5-1 | Validation Report | 5-2 |
| 6-1 | RID File Example | 6-42 |
| 7-1 | ReadmeSiv.txt | 7-2 |
| 7-2 | Configuration File Example | 7-15 |
| 7-3 | SIV Session Starup Messages | 7-17 |
| 7-4 | Auto-Tester Script Example | 7-24 |

# LIST OF TABLES


| Table | Title | Page |
|---|---|---|
| 1-1 | Operator Directive Syntax Notation | 1-5 |
| 2-1 | SIV Operator Directives | 2-1 |
| 3-1 | SIV Displays | 3-1 |
| 5-1 | SIV Reports | 5-1 |
| 6-1 | Rapid Interface Definition File Layout | 6-2 |
| 6-2 | Transmission Record Format | 6-2 |
| 6-3 | Field Record Format | 6-3 |
| 6-4 | Raw Data Record Format | 6-3 |
| 6-5 | Value Representation Formats | 6-3 |
| 6-6 | Transmission Record Keywords | 6-4 |
| 6-7 | Field Record Data Types | 6-12 |
| 6-8 | Field Record Actions | 6-13 |

# 1

# OVERVIEW AND OPERATING MODES

## 1.1 Overview and Operating Modes

Program Set Name: Subsystem Interface Verifier
Identification No.: DOI-5249-TP
Version No.: B
Software Release: 3.1 (1.3.1)
Pseudonym: SIV
Project: DSN
Subsystem: DSN Operations Interfacing Software (DOI)
Assembly: SIV

## 1.2 Overview

The Subsystem Interface Verifier (SIV) provides a means to simulate and test subsystem interfaces as defined in DSN interface agreements (820-16 and 820-13). Such interfaces are initially limited to those defined for data transmissions on the DSCC SPC LAN using the DSN standard 890-131 and DSN Network Data Flow Standard (DFL-1-1) protocols.
The SIV is used to create bit-level interface data, essentially prototyping an interfacing subsystem without the expense of creating special simulation code for that subsystem. It can also receive data from a subsystem, dump it in a readable format, and validate the contents. This allows interface testing to be performed and errors corrected as soon as interface agreements reach Level 3. Benefits of the SIV include early discovery of interface problems, decreased cost and complexity of interface testing, and improved schedule performance.

The SIV is being developed and delivered in successive builds and initiated with a prototyping effort. Customer demonstrations are held to demonstrate functionality of the SIV prior to delivery of each build. Each build focuses on increased functionality and enhancements to support subsystems under development which would best benefit from use of the SIV in the corresponding time frame.

## 1.3 Document Scope

This manual serves as a reference to using the Subsystem Interface Verifier (SIV) and its support tools, the Interface translator and the RID editor. The document is divided into two components to provide

1. a Software Operator's Manual (SOM) to using SIV and its support tools operationally, (i.e., directives, displays, messages) and

2. a User's Guide (UG) to assist in preparation for subsystems interface testing (i.e., connectivity, data files, automated and manual operating modes).

## 1.4 Equipment Setup

SIV and SIV support tools run on SUN Solaris platforms and can operate either in the user's development lab or at DTF-21. Section 7, USERS GUIDE, details the SIV connectivity and operating modes available and equipment and system setup instructions.

The SIV support tools, the Interface translator and the RID editor, are based only on UNIX functionality and, therefore, can presumably operate on other UNIX platforms with a minimal port effort.

## 1.5 Conventions and Notations

### 1.5.1 Conventions

All lists in this manual are shown in alphabetical order. The following conventions are followed in this manual for the user interfaces (e.g., Operator interactions, menus, forms, etc.) produced by SIV. Each user interface is presented with the following information:

a. description of the purpose of the user interface
b. additional information (including itemized description of contents and data)
c. limitations and notes
d. responses and rejections

The following conventions will be used for character representation on sample user outputs presented in this manual:

| | |
|---|---|
| **ddd** | Day of Year (DOY) |
| **hh** | Hours (GMT) |
| **mm** | Minutes (GMT) |
| **ss** | Seconds (GMT) |
| **a** | alphanumeric character |
| **n** | decimal integer |
| **x** | hexadecimal character |

User outputs (displays, reports, forms, and menus) are characterized by their size and dynamics.

#### 1.5.1.1 Operator Directives Conventions

The ':' character is specified as an optional first parameter on many of the OD syntax forms. The ':' is required in those cases when the syntax may be in violation of the standards established in 890-133. If the ':' were not included, the directive would be rejected by the LMC.

#### 1.5.1.2 Display Conventions

Size refers to amount of area that is occupied on a screen. Size is either Half screen (41 columns), or Full screen (85 columns) in width.

Dynamics refers to how a display screen area is accessed. There are two types of dynamics: **scrollable and non-scrollable**. **Scrollable** is used when there are more lines of data than can be presented in the screen area at one time. The excess data may be viewed by using the Line Up, Line Down, Page Forward and Page Backward scroll keys at the DMC console. Scrollable displays can also be controlled with the VIEW OD. Scrollable displays are denoted by a vertical up/down arrow just to the left of the day field in the title line. The arrows indicate the directions in which more data exists. If the up arrow is present, the display can be scrolled with Line Down and Page Backward. If the down arrow is present, the display can be scrolled with Line Up and Page Forward. **Non-scrollable** is used when all of the data is present within the screen area.

#### 1.5.1.3 Message Conventions

SIV produces the following types of messages **Alarm, Advisory, Prompt, Log Only,** and **Other**. All alarm and advisory messages received or generated by SIV are recorded on a log that can be displayed on the screen in a scrolling manner.

Messages produced by the SIV are grouped by category. Within each category the messages are listed alphabetically is Section 4.

**Alarm** messages are about abnormal conditions for which the operator is expected to take some corrective action, or to be aware that some corrective action is being carried out automatically. There are three levels of alarm message severity: *Emergency*, *Critical*, and *Warning*. *Emergency* level indicates an immediate danger to personnel or facilities. *Critical* level requires operator action and operations may degrade further. *Warning* level does not require operator action to prevent operations from degrading further. *Critical* level alarm messages are displayed in red in the alarm section on the screen, *Warning* level alarm messages are displayed in yellow, and the resolving of alarm conditions (clearing) in white.

**Advisory** messages inform the operator of minor malfunctions, changes in status, routine progress, etc. Operator action is not required in response to an advisory message.

**Prompt** messages instruct the operator to perform a specific action.

**Log Only** messages reflect conditions which do not affect SIV operation and can typically be ignored by a user. For example, Log Only messages can be used for detailed progress, to further identify parameters in an already reported software anomaly, etc. These can be viewed through access to the terminal Log file.

**Other** messages include general response and rejection messages in reply to Operator Directives, and appear to the operator on the message line (line 20). Specific response or rejection messages are listed as part of the detailed descriptions for Operational Directives.

The SIV response to an Operator Directive has four general forms. If the input is valid and no procedural errors have been made, SIV will respond with 'COMPLETED' or 'STARTED'. If the input is unrecognizable, invalid, or a procedural error has been made, SIV will respond with 'REJECTED' and include a message detailing the error. If the input is a single or double question mark ("?" or "??"), the SIV will respond with the a brief description of the OD or the OD's synopsis (syntax), respectively.

### 1.5.2 Notation

#### 1.5.2.1 Operator Directive Syntax Notation

The syntax of each operational directive is specified using a modified Backus-Naur Form (BNF) grammar.

Table 1-1 Operator Directive Syntax Notation

| NOTATION | DESCRIPTION |
|---|---|
| ALLCAPS | Any capital letters, integers or punctuation marks (other than those required for Backus-Naur presentation) must be input exactly as specified. |
| <parameter> | Lower case letters between angle brackets are used to indicate that the angle brackets and text in between them must be replaced with some other text as specified in the accompanying notes to the directive format. |
| \| | The "or" bar indicates that either the symbol to the left of the bar or the symbol to the right of the bar must appear, but not both. If a choice must be made from more than two, the possible choices will appear in sequence with bars separating each. The bar is not input. |
| [OPTION] | Anything that appears between square brackets may be input or blanks may be input in its place. The brackets are not input. |
| {1 \| 2} | The braces are used to group symbols together for clarity. The braces are not input. |
| <value>... | When a symbol or group of symbols is followed by three periods, this indicates that the symbol may be repeated up to the specified number of times. |

| | |
|---|---|
| | When a blank space appears between symbols, one or more blank spaces or a comma may be input. When no space appears between symbols, the directive must be entered with no space at that point. |

### 1.5.2.2 Display Notation

SIV provides a number of displays which operators may use to monitor the status of the subsystem and to aid in the system's configuration. SIV supports a maximum of twelve simultaneous displays. Displays are accessed via requests from the DMC or local terminal.

Line 1 of each display is the header. The header consists of the source's Directive Destination Code (DDC), mnemonic, title, day of year, and time of day in hours, minutes, and seconds.

The following conventions will be used for the DMC Display Request format to request SIV displays at the DMC console:

**<ddc> D aaaaa nn [rr]**

where:

| | | |
|---|---|---|
| **<ddc>** | - | Directive Destination Code for SIV |
| **D** | - | Display directive (D is for display) |
| **aaaaa** | - | Display mnemonic (5 characters) |
| **nn** | - | Screen region on which the display is to appear (acceptable values are 11, 14, 31-34 and 41-44) |
| **rr** | - | Optional refresh rate in seconds (default is 5), range: 1-99 |

At the local terminal, the same syntax applies, except the DDC is not required. A list of valid display names will appear on the requested region instead.

### 1.5.2.3 Message Notation

Messages are shown in this manual just as they are output by the software, except that variable fields are shown by triplets of lower case letters (e.g., aaa or fff). The accompanying descriptive text will explain the content and meaning of the variable fields. The response and rejection messages for each OD are documented in Section 2. All other SIV messages are documented in Section 4.

### 1.5.2.4 Example Notation

Examples are shown in this manual using the courier font:

example3

## 1.6 Controlling Document

| | |
|---|---|
| D-4006 | JPL Software Management Standards Package: Software Design Phase; Version 3.0, December 1988 |
| D-4007 | JPL Software Management Standards Package: Implementation and Test Phase; Version 3.0, December 1988 |

## 1.7 Applicable Documents

| | |
|---|---|
| | C - A Reference Manual; S.P. Harbison & G.L. Steele; Prentice-Hall, Inc.; 4th Edition, 1995 |
| UG-DOI-5527-OP | Multi-Use Software - Shared Software UG/SOM; Revision A |
| UG-DOI-5484-OP | Mission-Dependent Equipment Dsn Network-Level Data-Flow Common Software - 201 Us¨rs Guide |
| 820-13 | DSN System Requirements - Detailed Interface Design; Revision A, Release 209, February 1993 |
| 820-16 | DSN System Requirements - Mark IVA Detailed Subsystem Interface Design; Release 71, November 1992 |
| 820-19 | DSN System Requirements - Detailed Interface Design Standard; August 1992 |
| 820-19 DFL-1-1 | Network-Level Data Flow Standard; August 1992 |
| 820-19 DFL-1-2 | DSCC General Data Flow Standard; March 1994 |
| 820-19 DFL-1-3 | DSN Requirements for the USE of the TCP/IP Internet Protocol Suite; October 1994 |
| 820-19 DFL-1-4 | NOCC Communication Standards; July 1994 |
| 890-131 | DSCC General Data Flow Standards; Revision C, Release 3, July 1992 |
| 890-132 | DSCC Monitor and Control Data Interchange Standards; Revision C, Release 1, May 1991 |
| 890-133 | Man-Machine Interface Standards for Subsystems Controlled by the DMC; Revision B, February 1991 |
| 890-134 | DMC Operator Interface Description; January 1984 |

# 2

# OPERATOR DIRECTIVES

## 2.1 Operator Directives - Quick Reference

More detailed information on the Operator Directives (OD) is in Section **Error! Reference source not found.**, except for the Multi-Use Software (MSW) ODs which are documented in Section 2 of the MSW SOM. (The MSW ODs are listed as convenience and are not completely described here.)

Table 2-1 SIV Operator Directives

| OD | DESCRIPTION | SYNTAX | XREF |
|---|---|---|---|
| ACTL | Run Auto-Tester Scripts | ACTL<br>ACTL <name> [NOSUSP] [<parameter>...]<br>ACTL RESM [OD \| EVT \| WAIT] [<parameter>...]<br>ACTL { SUSP [E \| D] } \| END | MSW |
| CNF | Configure the SIV | CNF<br>CNF <target> [NOFAT] [CDIR=<config-directory>] [RDIR=<rid-directory>] [TADD=<target-address>] [SADD=<siv-address>] [LINK=<link-no>] | 2-6 |
| D | Start a Display on Screen n in Quadrant m<br>Scroll a Display (displays are listed in | D <display> { <n><m> [REFRESH] } \| OFF<br>D <display> {U \| D \| F \| B} | MSW |

| OD | DESCRIPTION | SYNTAX | XREF |
|---|---|---|---|
| | sect. 3) | | |
| D201 | Initialize the 890-201 Stream Routing Tables | D201<br><br>D201 {GEN \| RCV} | 2-9 |
| DFILE | Specify the ISB Dump File | DFILE {<file> \| STDOUT \| NULL} [RAW={<rawfile> \| OFF}] | 2-11 |
| DUMP | Dump ISB Message Blocks | DUMP {IN \| OUT} <mid> {E \| D} | 2-12 |
| ENOFF | Purge Off-line Messages File | ENOFF PURGE | MSW |
| (exit) | exit SIV - see TERM | TERM ABORT | MSW |
| FACT | Modify the Action and Values of a Field<br><br>in the default stream - see SDFLT | FACT <field> [<printf-format>[.<printf-format>...]]<br><br>FACT <field> <action> [<value>...] | 2-13 |
| FVAL | Modify the Values of a Field<br><br>in the default stream - see SDFLT | FVAL <field> [<printf-format>]<br><br>FVAL <field> <value>... [<scanf-format>] | 2-16 |
| IRT | Dump the Inbound Routing Table | IRT | 2-18 |
| LDP | Add or Delete Logical Data Paths | LDP<br><br>LDP {ADD \| DEL} <src-process-code> <dest-process-code>... | 2-19 |
| LOG | Log Inbound Streams | LOG [<stream>]<br><br>LOG {<stream> \| ALL} {E \| D} [HDR] | 2-21 |

<table>
<tr><th>OD</th><th>DESCRIPTION</th><th>SYNTAX</th><th>XREF</th></tr>
<tr><td>RAW</td><td>Specify the Raw Data File<br><br>for the default stream - see SDFLT</td><td>RAW<br><br>RAW <filename></td><td>2-23</td></tr>
<tr><td>RAWP</td><td>Change the Raw Data File Directory Path</td><td>RAWP<br><br>RAWP <directory></td><td>2-25</td></tr>
<tr><td>RCVB</td><td>Control Received Block Processing</td><td>RCVB { MID=<mid> | [MID=19] SEG=<segment-no> }<br><br>RCVB { ALL | MID=<mid> | [MID=19] SEG=<segment-no> } {E | D | RESET}</td><td>2-26</td></tr>
<tr><td>RCVM</td><td>Control Received Block Reporting</td><td>RCVM { MID=<mid> | [MID=19] SEG=<segment-no> }<br><br>RCVM {ALL | MID=<mid> | [MID=19] SEG=<segment-no> } C=<count></td><td>2-29</td></tr>
<tr><td>RIDP</td><td>Change the RID File Directory Path</td><td>RIDP<br><br>RIDP <directory></td><td>2-32</td></tr>
<tr><td>SCOM</td><td>Set the Communication Mode for SIV</td><td>SCOM</td><td>2-33</td></tr>
<tr><td></td><td></td><td>SCOM {Q | U | A} [SRC=<process-code>] [DST=<process-code>] [DDC=<ddc>]</td><td></td></tr>
<tr><td>SDFLT</td><td>Set the Default Stream</td><td>SDFLT<br><br>SDFLT <stream></td><td>2-35</td></tr>
</table>

| OD | DESCRIPTION | SYNTAX | XREF |
|---|---|---|---|
| SLOAD | Load RID into Stream Control Table | SLOAD <stream> [+AS \| +DE \| +HI \| +MI \| +LO] | 2-36 |
| SREM | Remove RID from Stream Control Table | SREM [<stream>] | 2-38 |
| SSEND | Send Stream Data Blocks to Destination | SSEND [<stream>] { [C=<count> \| M=<minutes>] [F=<delay>] [D=<p-c>] } \| END | 2-40 |
| STRM | Modify the 890-131 Stream Header | STRM [<stream>]<br><br>STRM [<stream>] [SRC=<process-code>] [DST=<p-c>] [MID=<mid>] [LAN=<lan>] [PROTO=<proto>] [SAC=<sac>] [F=<delay>] [C=<count>] | 2-42 |
| TERM | Initialize a Terminal Screen<br><br>(this is an incomplete list - see MSW SOM)<br><br>Execute a shell (ksh) command line | TERM SCREEN <n> ABORT<br><br>TERM SCREEN <n> [ANSI \| PSITECH \| TEK \| VT100 \| WY370 \| XPSI] <device><br><br>TERM {SHELL \| SPAWN} “ <command-line>“ | MSW |
| TGT | Send Operator Directive to the Target | TGT <subsystem-OD> | 2-44 |
| VAL | Validate the Logged Inbound Stream | VAL [<stream>]<br><br>VAL {<stream> \| ALL} {E \| D} [<block-count>] | 2-46 |
| VIEW | Scroll a Display | VIEW <display> {U \| D \| F \| B \| L \| P \| S} [<parameter>] | MSW |
| XPSI | Generate X11 Screens for the Subsystem | XPSI {XT1 \| XT2 \| XT3 \| XT4} | 2-48 |

DESCRIPTION This configures the SIV for testing the specified target's interfaces.

SYNTAX

CNF
CNF <target> [NOFAT] [CDIR=<config-dir>] [RDIR=<rid-dir>]

[TADD=<target-address>] [SADD=<siv-address>] [LINK=<link-no>]

where:

<target> DDC of the subsystem under test (1 to 4 characters).

NOFAT Specifies that SIV should not transmit FAT blocks.

<config-dir> Sub-directory containing the target's configuration file <target>.cnf . The default is ./cnfdir/.
<rid-dir> Sub-directory containing the target's interface definition files aaa.rid. The default is ./riddir/<target>/.
<target-addr> The ethernet address, in 12 hexadecimal digits, of the machine running the subsystem under test. The default is in the configuration file <target>.cnf.
<siv-addr> The ethernet address, in 12 hexadecimal digits, of the machine running the SIV. The default is in the configuration file <target>.cnf.
<link-no> Link number is range from 1 to 8. The default is link 1.

EXAMPLE

CNF DGT
Configure the SIV to test the DGT interfaces.

CNF ACG NOFAT
Configure the SIV to emulate the ACG subsystem interfaces with control from the DMC instead of the SIV.

CNF BVR CDIR=home RDIR=temp TADD=08bf1c049e20
Configure the SIV to test the BVR interfaces. Use configuration file bvr.cnf in the sub-directory ./home and the RID files from the sub-directory ./temp. The Ethernet address of the target machine is 08bf1c049e20.

| | |
|---|---|
| **CNF** | TBS |

NOTES If the src, dst, or mid of a rid file is changed during a SIV session, you must re-issue the CNF directive.

Parameter overrides for CDIR, RDIR, TADD, and SADD are only in effect until the next CNF directive is entered.

When using the NOFAT parameter, you will need to set the logical data paths using the LDP directive. You may also need to set the communication mode using the SCOM directive to control the SIV from another console, such as the LMC.

LIMITATIONS The keywords CDIR and RDIR can only specify sub-directories of the current directory.

The file suffixes ".rid" and ".cnf" are required for the configuration and RID files, respectively.

The hexadecimal input cannot have a leading 0x (e.g. ABCDEF is ok, but 0xABCDEF is not).

RESPONSES COMPLETED. SIV CONFIGURED FOR aaa

The SIV is configured to simulate the interfaces to the subsystem aaa.

REJECTIONS REJECTED. aaa LOAD ERRORS ENCOUNTERED, SEE LOG

The table data in the subsystems CNF file could not be loaded. Typically, the aaa.cnf file does not exist. Either correct the spelling of the target id aaa, or specify the directory path of the aaa.cnf file via the CDIR keyword.
REJECTED. aaa ADDRESS bbb IS BAD
The ethernet address bbb for subsystem aaa is not a hexadecimal number.

REJECTED. DIRECTORY aaa DOES NOT EXIST

| | |
|---|---|
| **CNF** | TBS |

The CNF or RID directory path aaa does not exist.  Either edit the configuration file, or override the directory path with the CDIR or RDIR keywords.

| **CNF** | TBS |
|---|---|

DESCRIPTION This reinitializes the 890-201 stream routing tables.

SYNTAX D201
D201 {GEN | RCV}

where:

GEN | RCV Reinitialize the 890-201 services for generation or reception using the stream definition file (SDF) sdf_gen.txt / sdf_rcv.txt.

EXAMPLE D201 GEN
Reinitialize the 890-201 services for generation using the SDF sdf_gen.txt.

NOTES SIV starts up with the initialization of the 890-201 services for reception using the SDF sdf_rcv.txt..

The 890-201 services require the following files:

ec_table.txt - ENCAP table
ldf.txt - LDF file
liff.txt - LIFF file
sdf.txt - SDF (SIV copies the appropriate sdf_aaa.txt file to this file)

See the 890-201 Users Guide, UG-DOI-5484-OP.

LIMITATIONS SIV does not supportpsimultaneous 890-201 generation and reception.
In theory, D201 RCV and D201 GEN should set up the 890-201 for reception or generation, respectively, but this is not enforced and what actually happens is solely determined by the contents of the SDFs.

RESPONSES COMPLETED. 201 (DFL-1-1) IS IN THE aaa MODE

| | |
|---|---|
| **D201** | TBS |

Either the directive was successful putting SIV in the 890-201 mode aaa (defined by the SDF aaa_sdf.txt) or this is the current state when the OD is entered without parameters.

<u>REJECTIONS</u> REJECTED. BAD DATA TYPES

The directive contained an unexpected or misspelled parameter.

| **D201** | TBS |
|---|---|

DESCRIPTION This specifies the name of the ISB dump file.

SYNTAX DFILE {<file> | STDOUT | NULL} [RAW={<rawfile> | OFF}]

where:

<file> Print the ASCII ISB dump output to the file ./tstdir/<file>.
STDOUT Print the ASCII ISB dump output to your terminal.
NULL Close the ASCII ISB dump file.
<rawfile> Print the raw ISB dump output to the file ./tstdir/<rawfile>.
OFF Close the raw ISB dump file.

EXAMPLE DFILE predict.nss
Dump ASCII ISBs to the file predict.nss.

NOTES The ISB ASCII dump file contains a hexadecimal / ASCII mapping format from the UNIX utility od(1) and is not a raw image of the block. The ISB raw dump file is a raw image of the block.

LIMITATIONS The directory path can not be changed.

RESPONSES COMPLETED.

REJECTIONS DUMP FILE FOPEN ()
File could not be opened. Check the existence of the subdirectory and ensure that a legal filename was entered.

| **DFILE** | TBS |
|---|---|

DESCRIPTION This dumps inbound or outbound ISB blocks by the 890-131 message identifier.

SYNTAX DUMP {IN | OUT} <mid> {E | D}

where:

<mid> 890-131 ISB message identifier (MID). The hexadecimal range is [00, ff].
IN | OUT Select inbound or outbound ISBs.
E | D Select enable (start) or disable (stop) dump.

EXAMPLE DUMP IN 10 E
Starts dumping inbound 890-131 blocks with message identifier hexadecimal 10 (CCN).

NOTES Output (file or screen) and type (raw or ASCII) of the dumped ISB blocks is controlled by the DFILE directive.

LIMITATIONS Error conditions are not flagged. Re-enter if you suspect an error was made.

RESPONSES COMPLETED

REJECTIONS None.

| **DUMP** | TBS |
|---|---|

DESCRIPTION This modifies the action and values for the specified field in the default stream.

SYNTAX FACT <field> [<printf-format>[**.**<printf-format>...]]
FACT <field> <action> [<value>...]

where:

<field> The field name as specified in RID file. Fields can also be specified by their zero based index with !FLD<n>, where n is the field's index. (e.g. the third field would be !FLD2).

<action> The action used to determine the value of the field for each generation of the data block.

<value> The new values for the specified action.

<aaa-format> The format of the output for a query.

The parameters can be either a format strings or values associated with the given <action>. If the first parameter after the action is a format string, then an inquiry of the current values of the action for the field are formatted according to the format string. Multiple format values can format each value of the action by separating the printf(3) format strings with periods.

EXAMPLE FACT CURRMODE P3 Q U A
Sets the action and values for the field CURRMODE to P and Q, U, A, respectively.

FACT !FLD023
Report the action and values for the 24th data field.

| **FACT** | TBS |
|---|---|

FACT SDBLEN %x.%x
Report the action and values, as hexadecimals, for the field SDBLEN.

NOTES The default stream is specified with the SDFLT directive. The list of streams in the stream table can be viewed with the STRM display. The SIV RID display can be viewed for identifying fields and current values. See Section 6 for a full description of RID field records, actions, and values. The printf(3) format is ANSI STDC. The formats for printf(3) and scanf(3) are not the same. The format parameters are identified by a leading % sign.

LIMITATIONS Only the default stream (see SDFLT) may be modified or displayed. The format of the parameter <format> is not the same for FACT and FVAL ODs. The more stringent error checking performed on the RID files is unavailable when setting fields with FACT or FVAL. Extreme caution must be used with these directives as illegal or out of bound values will not produce any warnings and will produce unpredictable outputs. The scalar number format (number base) is determined by the format of the input number.

This is contrary to the logic used in the FVAL OD, but consistent with the RID files. See *Value Representation Formats* in section 6.

RESPONSES COMPLETED. aaa: ACT=bbb VALS=[ccc ...]
The action and values for the field aaa in the default stream has been set to bbb and ccc, respectively.

| | |
|---|---|
| **FACT** | TBS |

REJECTIONS REJECTED.  STREAM TABLE EMPTY
Use SLOAD directive to load an interface definition file into the stream table.

REJECTED.  FIELD NOT FOUND
See the RID display for the valid field names.

REJECTED.  ACTION REQUIRES aaa VALUES
The specified action requires aaa arguments.  See section 6.

| **FACT** | TBS |
| --- | --- |

DESCRIPTION — This modifies the values for the specified field in the default stream.

SYNTAX — FVAL <field> [<printf-format>]
FVAL <field> <value>... [<scanf-format>]

where:

<field> The field name as specified in RID file. Fields can also be specified by their zero based index with !FLD<n>, where n is the field's index. (e.g. the third field would be !FLD2).

<value> The new values for the specified action.

<aaa-format> The format of the output for a query or the input when values are specified.

The parameters can be both a format string and values used by the field's action. If the first parameter after the <field> is a format string, then an inquiry of the current values of the field is formatted according to the format string. If a format string follows a values list, then the format string must match the input value type. The format strings are printf(3) formats for outputs and scanf(3) strings for inputs, with addition of the format "%xs" for converting input strings to lower-case letters.

EXAMPLE — FVAL DAYOFYEAR 321 %d
Sets the value of DAYOFYEAR to 321 decimal.

FVAL !FLD023 %x
Reports the value of the 24th field in hexadecimal format.

FVAL SDSETNM DGT__NSS %zs
Sets the value of SDSETNM to the lower-case string "dgt__nss".

| **FVAL** | TBS |
|---|---|

NOTES The default stream is specified via the SDFLT directive. The list of streams in the stream table can be viewed with the STRM display. The SIV RID display can be viewed for identifying fields and current values. See Section 6 for a full description of RID field records and values. The printf(3) and scanf(3) formats are ANSI STDC. The formats for printf(3) and scanf(3) are not the same. The format parameters are identified by a leading % sign.

LIMITATIONS Only the default stream (see SDFLT) may be modified/displayed. The format of the parameter <format> is not the same for FACT and FVAL ODs. The more stringent error checking performed on the RID files is unavailable when setting fields with FACT or FVAL. Extreme caution must be used with these directives as illegal or out of bound values will not produce any warnings and will produce unpredictable outputs. The scalar number format (number base) is determined by the optional format parameter and not the format of the input number. This is contrary to the logic used in both the FACT OD and the RID files. The default input format is decimal.

RESPONSES COMPLETED. aaa = bbb
The values for the field aaa in the default stream has been set to bbb.

REJECTIONS REJECTED. FVAL ERROR, FLD aaa NOT FOUND
See RID display for the list of valid mnemonics.

| **FVAL** | TBS |
|---|---|

DESCRIPTION This dumps the inbound routing table.

SYNTAX IRT

EXAMPLE IRT

NOTES The IRT table contains the list of MIDs currently registered for by one or more task queues and lists the queue numbers through which a task will receive the block.

The queue name which corresponds to the queue number presented in this table can be identified through entry of the MSW ODs "DBG OD E" and then "Q <queue-no>".

SIV ODs which affect the entries in the inbound routing table are DUMP, and RCVB.

LIMITATIONS Only the file util.log will be created. Subsequent entries of IRT will append to this file.

RESPONSES COMPLETED. DUMPING IRT TABLE
Indicates that table dump to util.log has been initiated.

REJECTIONS None.

| **IRT** | TBS |
|---|---|

DESCRIPTION This adds or deletes logical data paths that connect SIV to the subsystems on the LAN.

SYNTAX LDP {ADD | DEL} <source> <destination>...

where:

<source> The process code in hexadecimal from which paths will be established.

<destination> The process code in hexadecimal to which paths will be established.

EXAMPLE LDP ADD 54 AE0

Add the logical data path from 0x54 (BVR) to 0xAE0 (ACG, link 1).

LDP DEL 9B0 C10 0

Delete the two logical data paths from 0x9B0 (MPA, link 1) to 0xC10 (CMC) and from 0x9B0 to 0 (LMC).

NOTES Must be entered whenever "NOFAT" parameter is entered for the CNF SIV directive. This allows communication between the target subsystem and SIV when SIV is used to emulate subsystems but is NOT operating as the DMC. For example, when DTF-21 SIV is brought into a link as another assembly (e.g., BVR) with the appropriate directives at the CMC, the LDP directive is entered at the SIV in order to establish any logical data paths needed to send or receive data (as if SIV were BVR).

The MSW display LDPST can be viewed for the logical data paths used by SIV.

LIMITATIONS The directive response message only reflects the first <destination> value associated with <source>.
The maximum number of logical data paths is 99. (This is a parameter to the stfmain task in the file sysinit.ini.)

| **LDP** | TBS |
|---|---|

RESPONSES COMPLETED. aaa bbb PC FROM 0xccc TO 0xddd
Reports the addition (aaa = ADD) or deletion (aaa=DEL) of bbb data paths for source ccc to destination ddd.

REJECTIONS REJECTED. aaa LDP INVALID bbb
The <source> (bbb=SRC) or the <destination> (bbb=DST) entered was invalid.

REJECTED. aaa LDP TABLE FULL
The number of logical data path has reached the maximum. Either remove unnecessary entries or request an extension of the maximum (see LIMITATIONS).

REJECTED. aaa LDP ENTRY NOT FOUND
The logical data path, <source> <destination>, you tried to delete does not exist.

REJECTED. aaa LDP UNDOCUMENTED LDP STATUS
The logical data path, <source> <destination>, you tried to add already does exist.

| **LDP** | TBS |
|---|---|

DESCRIPTION This controls the logging of inbound streams to file and is required for validation of the inbound streams.

SYNTAX LOG [<stream>]
LOG {<stream> | ALL} {E | D} [HDR]

where:

| | |
|---|---|
| <stream> | The stream name corresponding to one displayed in the CNF display (RID names w/o .rid). |
| ALL | Select all the streams listed in the CNF display. |
| E \| D | Select enable (start) or disable (stop) logging. |
| HDR | Include the 890-201 DDD header. (890-131 headers are never logged). |

EXAMPLES LOG otsnrt80
Query the current status of the stream otsnrt80.

LOG otstlm36 E
Begin logging the stream otstlm36 to file without the 890-201 DDD header.

LOG otsvsop E HDR
Begin logging the stream otsvsop to file and include the 890-201 DDD header.

NOTES The list of streams available for logging can be viewed with the CNF display. When a message is received, several internal parameters (see *Transmission Record Keywords* in Section 6) are compared with the values of the RID files, listed in the CNF display, for the *first* exact match. If logging is enabled for that stream, SIV logs the data to a file with the same name as the stream, but with the suffix ".log".

| **LOG** | TBS |
|---|---|

LIMITATIONS Only the first stream with the same comparison parameters as any other stream can be logged at any one time. For example, stream A and stream B both have logging enabled, have the same comparison parameters, and stream A is ahead of stream B in the CNF list. When stream B arrives, it will be logged as if it were stream A.

RESPONSES COMPLETED. LOGGING aaa FOR bbb
Logging, without 890-201 header, is enabled (aaa=E) or disabled (aaa=D) for stream bbb.

COMPLETED. LOGGING aaa FOR bbb (INCLUDING HDR)
Logging, with the 890-201 header, is enabled (aaa=E) or disabled (aaa=D) for stream bbb.

COMPLETED. LOG E for aaa STRMs, LOG D for bbb STRMS
Logging is enabled for aaa number of streams and is disabled for bbb number of streams.

REJECTIONS REJECTED. NO STREAM DEFINITION FOR aaa
The stream aaa is not in the list of RIDs in the CNF display.

| **LOG** | TBS |
|---|---|

DESCRIPTION This specifies the name of a raw data file for the default stream.

SYNTAX RAW
RAW <filename>

where:

<filename> The name of a file containing binary or ASCII data. The file has the naming convention <name>.raw, where name has a 8 character maximum length, it must follow the normal naming conventions of the disk operating system, and all the letters must be *lower-case*.

EXAMPLE RAW delay.raw
Change the raw data file in the default stream to the file delay.raw.

NOTES This directive only works on the default stream (see SDFLT) and the default stream must be a raw data stream (see *Raw Data Record Description* in section 6).

LIMITATIONS The raw file directory defaults to the current directory, but can be changed with the RAWP directive.

RESPONSES COMPLETED. aaa SIZE=bbb SEGS=ccc
The default stream will contain bbb words of raw data from the raw data file aaa for each transmission. It will require ccc transmissions to exhaust the data (up to EOF) in file aaa.

REJECTIONS REJECTED. FILE NAME aaa DOES NOT EXIST
Use the RAWP directive to specify the directory path to aaa.

REJECTED. STREAM TABLE EMPTY.
Load RID files with the SLOAD directive.
REJECTED. NOT A RAWFILE RID.

| | |
|---|---|
| **RAW** | TBS |

The default stream is not a raw data stream.

REJECTED.  STREAM IS NOT IDLE.
Stop data generation with the SSEND directive before changing raw files.

| **RAW** | TBS |
|---|---|

DESCRIPTION This changes the raw data file directory path.

SYNTAX RAWP
RAWP <raw-dir>

where:

<raw-dir> The new directory path where the raw data files can be found.

EXAMPLE RAWP rawdir/dgt

NOTES See RAW and *Raw Data Record Description* in section 6.

LIMITATIONS The directory path is not validated by this directive. Invalid paths are reported when loading a raw data file with RAW or SSEND directives. The path cannot be more than 66 characters*, but this is not checked for*.

RESPONSES COMPLETED. RAWPATH=aaa
The new path to raw rata file directory is aaa.

REJECTIONS None.

| **RAWP** | TBS |
|---|---|

DESCRIPTION This controls which blocks inbound to SIV will be processed by SIV and can reset the block counts to zero.

SYNTAX RCVB {MID=<mid> | [MID=19] SEG=<seg-no>}
RCVB {ALL | MID=<mid> | [MID=19] SEG=<seg-no>} {E | D | RESET}

where:

<mid> The message ID (MID) number in hexadecimal for the block type.

<seg-no> The monitor data (MD) segment id (only when MID=19).

E | D Select enable (start) or disable (stop) the block reception.

RESET Reset the block's reception count to zero.

EXAMPLE RCVB MID=14
Query status of reception for support data (SD) blocks (MID=14).

RCVB MID=19 D
Turns block reception off for all monitor data (MD) blocks (MID=19).

RCVB MID=19 SEG=1 RESET
RCVB SEG=103 RESET
Resets the block reception count of MD blocks with segment id 1 or 103 to zero.

RCVB ALL ON
Turns on block reception for all blocks.

NOTES The RESET parameter does not differentiate as to whether reception for a given block type (MID) is E or D.

| **RCVB** | TBS |
|---|---|

Certain data block types are initially set for reception at SIV startup by MID: error blocks (MID=0); 201 blocks (MID=02, 04, or 05); CCNs (10); ODs and OD responses (12, 13); Support Data and SD Responses (14, 15); ENs (16); Monitor Data & MD Responses (19, 18); DGCs and Displays (1A, 1B); MD Self-Identifying Protocol (MIDs=30, 31, 32, 33); and FAT Updates & Locks (F0, F5).

LIMITATIONS Data blocks are not differentiated by source nor destination code for RCVB reception controls. Entry of the SEG parameter results in MID defaulting to hexadecimal 19 (Monitor Data).

RESPONSES COMPLETED. ALL BLOCK COUNTS RESET TO 0
Data block counters for all block types and all monitor data segments have been reset to zero.

COMPLETED. RCV aaa FOR ALL BLOCKS, COUNTS RESET TO 0
Reception for all blocks has been turned on (aaa=E) or off (aaa=D) and the block counters reset to zero.

COMPLETED. RCV aaa FOR ALL BLOCKS, COUNTS UNCHANGED
Reception for all blocks has been turned on (aaa=E) or off (aaa=D).

COMPLETED. RCV aaa FOR bbb BLOCKS MID=ccc BLKCNT=ddd
Reception processing is on (aaa=E) or off (aaa=D) for bbb (a textual description for the block type as identified by MID number). The MID number is also shown in hexadecimal (ccc) as well as the current block count ddd.

COMPLETED. RCV aaa FOR MONITOR DATA BLOCKS MID=19 SEG=bbb BLKCNT=ccc
Reception processing is on (aaa=E) or off (aaa=D) for MD blocks with segment id bbb and the count is ccc.
RCV aaa->bbb ccc CNT=ddd
Reception processing is on and a new block was received, where aaa is the source, bbb is the destination, ccc is the block type, and ddd is the block count.

REJECTIONS REJECTED. CSREG/CSDREG ERROR, SEE LOG.

| | |
|---|---|
| **RCVB** | TBS |

MSW service has reported an error in attempting to register or deregister a block for reception control. An explanatory message has been printed to the event message log.

REJECTED. QUERY MODE AVAILABLE WITH MID= OR SEG= PARMS
A query was attempted without a parameter.

REJECTED. INVALID SEG=aaa, VALID RANGE: 1 TO 999
The <seg-no> parameter (aaa) entered was not within the valid range defined for Monitor Data segments.

REJECTED. SEG VALID FOR MD ONLY (MID=19)
The <seg-no> parameter was entered with a MID which was not 19 (i.e., not Monitor Data). Only Monitor Data blocks (MID=19) can be further qualified with a segment number.

| **RCVB** | TBS |
|---|---|

DESCRIPTION This controls reporting of block reception, setting the frequency of the block reception Event Message (RCV).

SYNTAX RCVM {MID=<mid> | [MID=19] SEG=<seg-no>}
RCVM {ALL | MID=<mid> | [MID=19] SEG=<seg-no>} C=<count>

where:

<mid> The message ID (MID) number in hexadecimal for the block type.
<seg-no> The monitor data (MD) segment id (only when MID=19).
<count> The number of blocks to receive before the RCV Message is output. The range is [0, 32767]. A zero count suppresses the message completely.

EXAMPLE RCVM MID=14
Query current message reporting block count for Support Data (MID=14) blocks.

RCVM MID=19 C=5
One RCV message will be printed for every 5 Monitor Data (MID=19) blocks with the same segment id.

RCVM MID=19 SEG=103 C=100
RCVM SEG=103 C=100
One RCV message will be printed for every 100 Monitor Data (MID=19) blocks with the segment id 103.

RCVM MID=19 SEG=1 C=0
No RCV messages will be printed for any Monitor Data (MID=19) blocks with the segment id 1.

RCVM ALL C=0
No RCV messages will be printed at all.

NOTES The default message reporting frequency at SIV initialization is one (RCVM ALL C=1). RCVM only controls reporting of blocks received. RCVB controls the block reception processing.

| | |
|---|---|
| **RCVM** | TBS |

LIMITATIONS Data blocks are not differentiated by source nor destination code for RCVM reporting controls. Entry of the SEG parameter results in MID defaulting to hexadecimal 19 (Monitor Data).

RESPONSES COMPLETED. ALL BLOCKS, RCV MSG FILTER SET TO ccc BLKS
The receive message frequency (filter) counter has been set to ccc for all block types.

COMPLETED. MID=aaa, RCV MSG FILTER SET TO ccc BLKS
The receive message frequency (filter) counter has been set to ccc for block types with a MID of aaa.

COMPLETED. MID=19, SEG=bbb, RCV MSG FILTER SET TO ccc BLKS
The receive message frequency (filter) counter has been set to ccc for monitor data (MID=19) blocks with a segment id of bbb.

RCV aaa->bbb ccc CNT=ddd
The RCV message, where aaa is the source, bbb is the destination, ccc is the block type, and ddd is the block count.

REJECTIONS REJECTED. MUST SPECIFY MSG REPRESSION COUNT
The ALL parameter was specified without the C= parameter. No query is available for ALL.

REJECTED. INVALID SEG=aaa, VALID RANGE: 1 TO 999
The <seg-no> parameter (aaa) entered was not within the range defined for Monitor Data block segment ids.

| **RCVM** | TBS |
|---|---|

REJECTED.  SEG VALID FOR MD ONLY (MID=19)
The <seg-no> parameter was entered with a MID which was not 19.  Only Monitor Data blocks (MID=19) can be qualified with a segment id number.

REJECTED.  'SEG=' PARM MUST BE ENTERED FOR MD (MID=19)
The <seg-no> parameter must be entered when Monitor Data is specified.

REJECTED.  QUERY MODE AVAILABLE WITH MID= OR SEG= PARMS
Query has been attempted without a parameter which can identify valid query information.

| **RCVM** | TBS |
|---|---|

DESCRIPTION This changes the RID file directory path.

SYNTAX RIDP
RIDP <rid-dir>

where:

<rid-dir> The new directory path where the RID files can be found.

EXAMPLE RIDP riddir/dgt

NOTES None.

LIMITATIONS The directory path is not validated by this directive. Invalid paths are reported when loading a RID file with SLOAD directives. The path cannot be more than 66 characters, *but this is not checked for*.

This directive does not change the path used by the CNF directive. As such, it will NOT be used for any stream logging or validation.

RESPONSES COMPLETED. RIDPATH=aaa
The new path to RID file directory is aaa.

REJECTIONS None.

| **RIDP** | TBS |
|---|---|

DESCRIPTION This sets the communication mode for SIV, enabling remote control of SIV from another subsystem (i.e. LMC).

SYNTAX SCOM
SCOM {Q | U | A} [SRC=<source>] [DST=<destination>] [DDC=<mnemonic>]

where:

Q | U | A The communication mode = Quiescent, Unassigned, Assigned.

<source> The source process code in hexadecimal of the subsystem SIV is emulating.

<destination> The destination process code in hexadecimal of the new controller (e.g. LMC).

<mnemonic> The subsystem DDC or 3-4 character mnemonic of the subsystem SIV is emulating.

EXAMPLE SCOM A SRC=930 DST=0 DDC=MDA
Set the SIV (configured as a MDA) to assigned - it can accept directives (SIV directives, not MDA directives) from the LMC, send the responses and SIV generated event messages back to the LMC.

NOTES SIV should be configured with the CNF directive before using the SCOM directive.
SIV cannot respond to CCN (Configuration Change Notification) blocks from the LMC. The SCOM directive handles the communication mode switch and remote control set-up ordinarily handled by the CCNs.

LIMITATIONS The SCOM directive does not set the logical data paths (LDP). Use the LDP directive to set the LDPs.
SCOM does not control data flows to other addresses. It only enables control flow (i.e. directive responses, event messages, real monitor data, etc.) and does not control simulated data such as those from the SSEND directive.

RESPONSES COMPLETED. COMM=aaa SRC=bbb DST=ccc DDC=ddd
Set or query entry accepted and these are the new or current values.

| | |
|---|---|
| **SCOM** | TBS |

REJECTIONS REJECTED.  MSW GET COMM MODE ERROR=(aaa)
Unable to retrieve the current values due to the MSW error aaa.  Any new values were not set.

| **SCOM** | TBS |
|---|---|

DESCRIPTION This sets the default stream used by stream oriented directives.

SYNTAX SDFLT
SDFLT <stream>

where:

<stream> The name of RID file without the ".rid" suffix.

EXAMPLE SDFLT GENCCN
Establishes the RID file genccn.rid as the default stream which will be used whenever a stream specification is omitted as a parameter in stream oriented ODs.

NOTES The specified <stream> must have previously been loaded into the stream table with the SLOAD directive.

The list of streams in the stream table can be viewed with the STRM display.

The stream oriented directives are FACT, FVAL, RAW, SREM, SSEND, and STRM.

LIMITATIONS None.

RESPONSES COMPLETED. DFLT STREAM=aaa
The default stream is aaa, where aaa corresponds to the interface definition file aaa.rid.

REJECTIONS REJECTED. STREAM NOT LOADED
See the STRM display for a list of loaded streams.

| **SDFLT** | TBS |
|---|---|

DESCRIPTION This loads a Rapid Interface Definition (RID) file into the stream control table.

SYNTAX SLOAD <stream> [+AS | +DE | +NI | +MI | +LO]

where:

<stream> The name of RID file without the ".rid" suffix.

+AS | +DE | +NI | +MI | +LO

This changes every Variable action field in the RID.

EXAMPLE SLOAD GENCCN
Loads the RID file genccn.rid into the stream control table.

NOTES Use the SREM directive to remove an entry before adding another if the table is full. The list of streams in the stream table can be viewed with the STRM display. The list of streams available for loading into the stream table can be viewed with the CNF display. See *Variable Action* under *Action Descriptions* in section 6 for a description of the +aaa parameters. SIV searches for the RID files in the directory specified in the configuration file, the directory specified by the RDIR parameter to the CNF directive, or the directory specified by the RIDP directive - usually ./riddir/<ddc>.

LIMITATIONS Only 6 streams can be loaded at a time.

There is an upper limit of 500 fields per RID when more than one RID is loaded. With only one RID loaded, the limit is 2999 fields.

RESPONSES COMPLETED. SLOAD OF aaa COMPLETE
The RID file aaa.rid has been loaded into the stream table without errors.

| **SLOAD** | TBS |
|---|---|

FIELD COUNT EXCEEDED 500. SREM BEFORE NEXT SLOAD.
This is a reminder that the RID exceeds the 500 field limit and cannot share the stream table with other RIDs.

REJECTIONS REJECTED. aaa RID ERRORS - USE RIDLINT ON RID.
The RID contains errors and cannot be loaded. Use ridlint to report the errors in detail.

REJECTED. FIELD COUNT EXCEEDED aaa.
If aaa = 500, you may be able to load this RID by itself. If aaa == 2999, the RID is too large.

REJECTED. riddir/aaa/bbb.rid: ccc.
The file riddir/aaa/bbb.rid could not be read. The error message ccc will tell you why.

REJECTED. STREAM TABLE FULL (MAX=6)
The maximum number of interface definition files in the stream table has been reached.

REJECTED. RID aaa ALREADY LOADED
The RID for stream aaa is already in the stream table.

| **SLOAD** | TBS |
|---|---|

DESCRIPTION This removes the specified Rapid Interface Definition (RID) stream from the stream control table.

SYNTAX SREM [<stream>]

where:

<stream> The name of RID file without the ".rid" suffix.

EXAMPLE SREM GENCCN
Removes the RID (defined in the file genccn.rid) from the stream control table.

SREM
Removes the default stream RID definition.

NOTES The SDFLT directive specifies the default stream.
The list of streams in the stream table can be viewed with the STRM display.

LIMITATIONS None.

RESPONSES COMPLETED. SREM OF aaa COMPLETE
The RID aaa has been removed from the stream table.

COMPLETED. NEW DFLT=aaa
The default stream RID has been removed from the stream table and the default stream has been changed to aaa.

REJECTIONS REJECTED. aaa NOT LOADED
The stream aaa isn't loaded. Check your spelling.

REJECTED. STREAM TABLE EMPTY

| | |
|---|---|
| **SREM** | TBS |

There aren’t any streams loaded.

REJECTED. STREAM IS FOR RCV
This stream is validating blocks. Enter “VAL <stream> END” or wait until validation has stopped.

REJECTED. STREAM IS NOT IDLE
This stream is generating blocks. Enter “SSEND <stream> END” or wait until generation has stopped.

| **SREM** | TBS |
|---|---|

DESCRIPTION This starts or stops the generation of the specified stream, transmitting the ISB blocks.

SYNTAX SSEND [<stream>] { [C=<count> | M=<minutes>] [F=<delay> ] [D=<destination>] } | END

where:

<stream> The name of RID file without the “.rid” suffix.

<count> The number of blocks to transmit. The default is 1. To transmit indefinitely, specify 0. The range is [0, 32767].

<minutes> The number of minutes to transmit. This can be used instead of <count>.

<delay> The delay (in seconds and accurate to a tenth of a second) between block transmissions. The default is 1.0. The range is [0.1, $10^6$).

<destination> The destination process code in hexadecimal.

END Stop transmitting the stream.

EXAMPLE SSEND GENCCN C=1 D=56
Start generation of the stream genccn, sending only one block to process code 0x56.

SSEND GENCCN END
Stop transmitting the stream genccn.

SSEND
Start generation of the default stream, using the <count>, <delay>, and <destination> values from the RID file.

| **SSEND** | TBS |
|---|---|

NOTES The SDFLT directive specifies the default stream. The stream must be already loaded into the stream control table with the SLOAD directive. The list of streams in the stream table can be viewed with the STRM display. The C=<count> and M=<minutes> parameters are mutually exclusive.

LIMITATIONS See the directive SLOAD for the stream table limitations. There is a bug where the <delay> occasionally fall to zero. The work-around is to keep trying until SIV corrects itself.

RESPONSES COMPLETED.
The generation of the <stream> has been initiated.

COMPLETED. SSEND HALT SET FOR aaa
The generation of stream aaa has been stopped with the END parameter.

REJECTIONS REJECTED. aaa NOT LOADED
The stream aaa is not in the stream table. Check your spelling or load the RID with the SLOAD directive.

REJECTED. aaa IN GEN STATE
The stream aaa is already generating blocks. Stop the generation with the END parameter or wait until finished.

| **SSEND** | TBS |
|---|---|

DESCRIPTION This changes the values of the specified 890-131 stream header.

SYNTAX STRM [<stream>]
STRM [<stream>] [SRC=<source>] [DST=<destination>] [MID=<mid>] [LAN={ANY|SPC|HR}] [PROTO={PAR|NAP|DP}] [SAC=<sac>] [F=<delay>] [C=<count>]

where:

<stream> The name of RID file without the “.rid” suffix..

<source> The source process code in hexadecimal.

<destination> The destination process code in hexadecimal.

LAN The 890-131 LAN.
ANY any LAN
SPC SPC LAN
HR High Rate LAN
PROTO The 890-131 protocol.

PAR positive acknowledgment protocol, retransmits up to 3 times if acknowledgment is not received. Requires a Logical Data Path between source process code and destination process code on both the sending and receiving subsystems.

NAP no acknowledgment protocol, do not check for acknowledgment. Requires a Logical Data Path between source process code and destination process code on both the sending and receiving subsystems.

DP direct protocol, do not check for valid Logical Data Paths.
<sac> The 890-132 special application code. Values vary by subsystem interface agreement.

| **STRM** | TBS |
|---|---|

<mid> The 890-131 message id number in hexadecimal.

<delay> The number of seconds between generated block transmissions.

<count> The number of blocks to transmit. Zero indicates to send indefinitely.

EXAMPLE STRM GENMD11 DST=56 LAN=SPC PROTO=PAR MID=19 SAC=0
Override the genmd11.rid file values for the 890-131 stream header.

STRM GENCCN
Display current 890-131 header values for the stream genccn.

NOTES The SDFLT directive specifies the default stream.
The stream must be already loaded into the stream control table with the SLOAD directive. The list of streams in the stream table can be viewed with the STRM display.

LIMITATIONS Values are in effect until the RID file is loaded again via the SLOAD directive. Once re-loaded, the defaults specified in the RID are reinstated.

RESPONSES COMPLETED. aaa bbb/ccc F=ddd C=eee fff/ggg
The current configuration of stream aaa. Where bbb = source process code, ccc = destination process code, ddd = delay between blocks, eee = number of blocks to transmit, fff = the LAN (ANY, HR, or SPC), ggg = the LAN protocol (NAP, PAR, DP).

REJECTIONS REJECTED. aaa NOT LOADED
The stream aaa is not in the stream table. Check your spelling or load the RID with the SLOAD directive.

DESCRIPTION Forward the subsystem directive to the target subsystem packaged as if entered at the LMC.

| **TGT** | TBS |
|---|---|

SYNTAX TGT <subsys-OD>

where:

<subsys-OD> valid directive and associated OD parameters of the target subsystem. The OD syntax must conform to 890-132 in order to parse subsystem OD from OD parameters.

EXAMPLE TGT RUN NORPT
The directive RUN with parameters NORPT is sent to the target subsystem currently configured by SIV.

TGT D STS OFF
This will shut off the display block generation at the subsystem for the STS display (see LIMITATIONS).

NOTES The target is identified by the DDC within with the configuration file, loaded with the CNF directive.This uses the LMC process code if a RID is found containing the LMC process code; otherwise it uses the CMC process code if a RID is found containing the CMC process code. If neither is found, this directive will fail.

LIMITATIONS Displays requests are not supported. However, display requests can be tested for subsystem generation. The SIV will acknowledge the reception of display blocks at the SIV terminal, but without generation of a display.

RESPONSES COMPLETED. aaa
Directive was transmitted and completion OD response (aaa) from the target subsystem received.

REJECTIONS REJECTED. TGT TIMEOUT!
Directive was transmitted but no response was received. Check the logical data path and TGT communication mode (i.e., quiescent, unassigned, assigned).

| **TGT** | TBS |
|---|---|

DESCRIPTION This validates the specified inbound data stream.

SYNTAX VAL [<stream>]
VAL {ALL | <stream>} {E | D} [<count>]

where:

<stream> The name of RID file without the ".rid" suffix.

ALL Select all the streams.

E | D Either enable (start) or disable (stop) validation.

<count> The number of blocks to validate - the default is all that *have* arrived.

EXAMPLE VAL cmcseg01
Query current validation status for the stream cmcseg01.

VAL cmcseg99 E
Turn validation on for the stream cmcseg99.

NOTES The stream must have first been logged (see LOG directive) to file to validate. The stream must be already loaded into the stream control table with the SLOAD directive. The list of streams in the stream table can be viewed with the STRM display. Each invocation of VAL appends to the validation report file <stream>.val.

LIMITATIONS See the directive SLOAD for the stream table limitations. Validation is automatically disabled when the end of the log file is reached. Validation is reasonably fast, so unless you wish to be typing "VAL <stream> E" constantly, wait until after you have received all the blocks.

| | |
|---|---|
| **VAL** | TBS |

<u>RESPONSES</u> COMPLETED. VALIDATION aaa FOR bbb
Validation is enabled (aaa=E) or disabled (aaa=D) for stream bbb.

COMPLETED. VAL E FOR aaa STRMS, D FOR bbb STRMS
Query response summarizing the number of streams turned on and off for validation.

COMPLETED. VALIDATION OFF FOR ALL STREAMS
Validation control successfully set to OFF for any streams that were in progress.

COMPLETED. VALIDATION ON FOR ALL STREAMS
Validation turned on for all streams current in the stream control table.

<u>REJECTIONS</u> REJECTED. NO STREAMS CATALOGED, ENTER 'CNF' OD
Configure SIV with the CNF directive.

REJECTED. NO STREAM DEFINITION FOR 'aaa'
Load validation streams with the SLOAD directive.

| **VAL** | TBS |
|---|---|

DESCRIPTION Indirectly generate X11 screens for the subsystem on the SIV machine.

SYNTAX XPSI {XT1 | XT2 | XT3 | XT4 }

where:

XT1 | XT2 | XT3 | XT4

The name of the pipe (SYSV UNIX FIFO). in the current working directory where the X11 screen will read it's input from.

EXAMPLE XPSI XT2
This will create an X11 screen just like "TERM SCREEN 2 XPSI :0", except for reading input from ./XT2.

NOTES For this to work at all, the subsystem's display blocks must be written to the FIFO.

LIMITATIONS We don't know how this works and the author has departed...

RESPONSES COMPLETED. STARTED XPSI AT aaa
A XPSI screen was started, reading from the FIFO aaa.

REJECTIONS None.

## 2.2 Menus Hierarchy

SIV does not have any menus.

## 2.3 Menus - Detailed Description

SIV does not have any menus.

## 2.4 Icons

SIV does not have any Icons.

## 2.5 Forms

SIV does not have any Forms.

## 2.6 Function Keys

SIV does not have any Function Keys.

## 2.7 Other Interactions

Not applicable.

**THIS PAGE INTENTIONALLY LEFT BLANK**

# 3

# MONITORING AND STATUS DISPLAYS

## 3.1 Display - Quick Reference

More detailed information on the Displays is in Section 3.3, except for the Multi-Use Software (MSW) Displays which are documen ted in Section 3 of the MSW SOM. (The MSW Displays are listed as convenience and are not completely described here.)

Table 3-1 SIV Displays

| DISPLAY MNEMONIC | DESCRIPTION | ASSOCIATED DISPLAYS | XREF |
|---|---|---|---|
| ACTL / ACTLM / ACTLN / ACTLR | Auto-Tester: Control / Messages / Results / Reports | | MSW |
| CNF | Subsystem Configuration | | 3-4 |
| ENOFF | Offline Event Notices | | MSW |
| FNCAP | General Introductory Help | | MSW |
| HDIR / HDIS / HEVT | Full-Screen Help: Operator Directives / Displays / Event Notices | | MSW |
| ISB | ISB Dump | | 3-6 |
| LDPST | Logical Data Paths (See SIV SOM, Section 2, LDP) | | MSW |
| QDIR / QDIS | Half-Screen (Quick) Listings: Operator Directives / Displays | | MSW |
| RID | RID Interface Control Table | | 3-8 |

| DISPLAY MNEMONIC | DESCRIPTION | ASSOCIATED DISPLAYS | XREF |
|---|---|---|---|
| ACTL / ACTLM / ACTLN / ACTLR | Auto-Tester: Control / Messages / Results / Reports | | MSW |
| STRM | Stream Status | | 3-10 |
| STS | SIV Status | | 3-12 |
| UPDAT | New Features (Update)  Help | | MSW |

## 3.2 Displays - Hierarchy

Not applicable.

**THIS PAGE INTENTIONALLY LEFT BLANK**

| CNF | Subsystem Configuiration Display |
|---|---|

## 3.3 Displays - Detailed Descriptions

```
         1         2         3         4         5         6         7         8
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345
LT CNF                   SUBSYSTEM CONFIGURATION DATA                   259 13:10:15

LINK 1             SPC LAN           H/R LAN            IP ADDR
 ACG (AE0):  00004b0a2da0
 SIV (D09):  00004b0a2dc0

 5  RIDS FOUND IN DIRECTORY:  ./riddir/acg

  #   RID FILENAME   SRC        DST       MESSAGE ID       201 DT  SUB-ID  CTRLS
001   bvrsg10n.rid   BVR(054)   ACG(AE0)  Mon Data  (019)          (103)    L H V
002      acgsd.rid   CMC(C10)   ACG(AE0)  Sup Data  (014)                   L V -
003     ccnacg.rid   CMC(C10)   ACG(AE0)  CCN       (010)                   - - -
004   cmcacg01.rid   CMC(C10)   ACG(AE0)  Mon Data  (019)          (001)    - - -
005   cmcacg10.rid   CMC(C10)   ACG(AE0)  Mon Data  (019)          (010)    - - -
006     nrtacg.rid   BVR(054)   SCP(010)  JPL-SDB   (005)          (041)    L H -
```

Figure 3-1  Subsystem Configuration Display

| CNF | Subsystem Configuiration Display |
|---|---|

| | | | |
|---|---|---|---|
| DESCRIPTION | This display shows the current configuration of the SIV. | | |
| NOTES | Size = Full Width        Dynamics = Scrollable | | |
| LIMITATIONS | None | | |
| DATA | Line 3 | **LINK** link number 1-8 | |
| | | **SPC LAN** | SPC LAN ethernet addresses |
| | | **H/R LAN** | High Rate LAN ethernet addresses |
| | | **IP ADDR** | unused |
| | Line 4 | | target DDC, process code, ethernet address (hexadecimal) |
| | Line 5 | | SIV DDC, process code, ethernet address (hexadecimal) |
| | Line 7 | | number of RID definition files found in directory ’xxx’ |
| | Line 9 | | stream information column header |
| | | **#** | stream index number |
| | | **RID FILENAME** | rid file loaded into this stream |
| | | **SRC** | DDC and process code (hexadecimal) of the source processor |
| | | **DST** | DDC and process code (hexadecimal) of the destination processor |
| | | **MESSAGE ID** | MID description and message number (hexadecimal) |
| | | **201 DT** | 890-201 SDB data type (decimal) |
| | | **SUB-ID** | 890-132 monitor data segment number or 890-201 data type 0x41 packet identifier (hexadecimal) |
| | | **CTRLS** | Current settings for logging, logging with 890-201 header, validation in progress. |
| | Lines 10-15 | | stream information for each RID read in for target |

| **CNF** | Subsystem Configuiration Display |
|---|---|

```
LT ISB                  ISB DUMP:   bvrsg10n                           259 13:10:15

 BLOCK DUMPED AT 101  22:42:56
 MID=0x019, SAC=0, SRC=0x054, DST=0xAE0, LAN=SPC PROTO=NAP
 BUFFER LENGTH IS: 258 bytes (129 WORDS.)
    0:  CE81 3231 0004 3038 3531 3330 3030 3030    ~~21~~0851300000
    8:  3030 4b61 4406 dccd 494e 00ae 494e 00ae    00KaD~~~IN~~IN~~
   16:  4944 4c45 4153 1fae 0000 0000 3f9c 28f6    IDLEAS~~~~~~?~(~
   24:  c309 6e14 bfbe b852 c347 2666 bed7 0a3d    ~~n~~~~R~G&f~~~=
```

Figure 3-2  ISB Dump Display

| **ISB** | ISB Dump Display |
|---|---|

DESCRIPTION Provides a 'snapshot' of an inbound or outbound data block for the current default stream.

NOTES Size = Full Width Dynamics = Scrollable

LIMITATIONS Only the "default" stream may be displayed. (Default stream can be changed using SDFLT OD). The most recent data block will be displayed based on display refresh rate.

DATA

| | | |
|---|---|---|
| Line 3 | **BLOCK DUMPED AT** | day of year (DOY) and time of day ISB was dumped |
| Line 4 | **MID** | 890-131 message identifier (hexadecimal) |
| | **SAC** | 890-131 special application code (decimal) |
| | **SRC** | source process code (hexadecimal) |
| | **DST** | destination process code (hexadecimal) |
| | **LAN** | LAN selection |
| | **HR** | High Rate LAN |
| | **SPC** | SPC LAN |
| | **ANY** | The HR LAN if available, otherwise the SPC LAN |
| | **PROTO** | 890-131 protocol |
| | **PAR** | positive acknowledgment required |
| | **NAP** | no acknowledgment required |
| | **DP** | direct protocol |
| Line 5 | **BUFFER LENGTH IS** | block length in bytes (and 16-bit words) |
| Lines 6-n | | data block contents present in hex and ASCII |
| | nnn: | 16-bit word offset for first entry this line |
| | FFFF (8 times) | 16-bit word value in hexadecimal for next 8 words in data block |
| | text | ASCII equivalent for 16 bytes (~ signifies non-printable character) |

| ISB | ISB Dump Display |
|---|---|

```
          1         2         3         4         5         6         7         8
 12345678901234567890123456789012345678901234567890123456789012345678901234567890012345
 LT RID                   INTERFACE CONTROL TABLE                       259 13:10:15

  FLD NAME      BITS  OFFS TYPE ACT #VALUES  NEXT-GEN-VALUE
    0 SEGID        7     0    U   C       1  75
    1 SEGLEN       9     0    U   C       1  39
    2 SERNO       16     0    I   +       3  21
    3 RAMPNO      16     0    I   +       3  22
    4 SCNID       32     0    C   C       1  99
    5 TIMETAG     96     0    T           0  (MONITOR DATA TIME FORMAT)
    6 TYPE        32     0    I   S       9  0.000000
    7 RAMPRATE    64     0    F   C       1  480.000000
    8 FREQSTART   64     0    F   +       3  2112000000.000000
    9 TIMEBIAS    32     0    I   C       1  50
   10 FREQBIAS    64     0    F   C       1  10.000000
   11 BAND        16     0    U   C       1  1
   12 PHCNT1      32     0    U   C       1  1
   13 PHCNT2      32     0    U   C       1  2
   14 PHCNT3      32     0    U   C       1  3
   15 RAMPTIME    96     0    T           0  (MONITOR DATA TIME FORMAT)
```

Figure 3-3  RID Interface Control Table Display

| **RID** | RID Interface Control Table Display |
|---|---|

DESCRIPTION This display shows the contents of the "default" RID (Rapid Interface Definition) file.

NOTES Size = Full Width Dynamics = Scrollable
See Section 6 for a full descriptions of RID data types, actions, and values.

LIMITATIONS RID must be loaded through use of SLOAD directive.
RID (stream) must be the default stream (SDFLT directive).
All numeric values are shown in decimal.
Fixed Point Scaled Integers are displayed as **X** and not **F<n>** as in the RID definition.
Offsets (OFFS) are usually zero, signifying this field immediately follows the preceding field.

DATA

| | | |
|---|---|---|
| Line 3 | **FLD** | field index number (starting from zero) |
| | **NAME** | field mnemonic from RID |
| | **BITS** | size of field in bits |
| | **OFFS** | offset of field from start of block (specified in bits, starting from zero) |
| | **TYPE** | data type |
| | **ACT** | action that determines the generated data value |
| | **#VALUES** | the number of internal values (debugging information only) |
| | **NEXT-GEN-VALUE** | data value in the next block transmitted |
| Line 4-n | | Field information for each entry in RID. |

| **RID** | RID Interface Control Table Display |
|---|---|

```
         1         2         3         4         5         6         7         8
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345
 LT STRM                            Streams Status  LINK 1               259 13:10:15

   RID                                    XMIT  LAST     TIME BLK  REQ  RAW
   NAME         SRC DST MID PROTO LAN SAC STATE GEN-TIME FREQ XFER BLKS FILE
 1 bvrsg10n     054 AE0 019  NAP  SPC 000 IDLE  22:33:50 10.00  10   20 support.raw
```

Figure 3-4  Stream Status Display

| **STRM** | Stream Status Display |
|---|---|

DESCRIPTION — This display shows current status of the data streams loaded.

NOTES — Size = Full Width    Dynamics = Non-scrollable

LIMITATIONS — Block counts do not increment when a generation of count=0 is specified (i.e., for indefinite transmissions).

DATA

| | | |
|---|---|---|
| Line 1 | **LINK** | Link number 1-8 |
| Lines 3-4 | **RID NAME** | name of loaded RID file |
| | **SRC** | source process code (hexadecimal) |
| | **DST** | destination process code (hexadecimal) |
| | **MID** | 890-131 message identifier (hexadecimal) |
| | **PROTO** | 890-131 protocol |
| | **PAR** | positive acknowledgment |
| | **NAP** | no acknowledgment |
| | **DP** | direct protocol |
| | **LAN** | LAN selection |
| | **SPC** | SPC LAN |
| | **HR** | High Rate LAN |
| | **ANY** | High Rate LAN if available, otherwise SPC LAN |
| | **SAC** | special application code (decimal) |
| | **XMIT STATE** | current status of stream |
| | **IDLE** | not being generated |
| | **GEN** | data blocks being generated |
| | **LAST GEN-TIME** | time of day when ISB was last transmitted |
| | **TIME FREQ** | delay between transmission of blocks in seconds |
| | **BLK XFER** | current count of blocks transmitted |
| | **REQ BLKS** | total number of blocks to be transmitted |

| STRM | Stream Status Display |
|---|---|

**RAW FILE** name of raw file with this stream

Lines 5-10 Stream information for up to six loaded streams.

```
              1         2         3         4         5
     12345678901234567890123456789012345678901234567890
 1   LT STS  SIV STATUS DISPLAY  259 13:10:15
 2
 3   --  TEST CONFIG  --  --  AUTOTESTER  --
 4   MODE:  CONFIGURED    ACTL : NONE
 5   TGT :  MDA           STATE: IDLE
 6   #I/F: 5    LINK 1
 7   SIV :  CMC (C10)     --   FILTERS    --
 8          BVR (054)     VAL:   OFF
 9                        RPT:   OFF
10
11
12
13
14   -- DATA GENERATE --  -- DATA RECEIVE --
15   STATE:    E          STATE:    E
16   STATUS:   ON         STATUS:   ON
17   STREAMS:  1          STREAMS:  ANY
18   AG-RATE:             AG-RATE:
19   LOGGING:  OFF        LOGGING:  OFF
20
21   VALIDATE: OFF        VALIDATE: OFF
22   VAL STRM:            VAL STRM:
23   LAST BLK:            LAST BLK:
24
```

Figure 3-5  SIV Status Display

| **STS** | SIV Status Display |
|---|---|

DESCRIPTION — This display shows the overall status of the SIV

NOTES — Size = Half Width    Dynamics = Non-scrollable

LIMITATIONS — Certain display values are not yet available.

DATA

| | | |
|---|---|---|
| line 3 | | **-- TEST CONFIG --** |
| line 4 | **MODE** | **STANDBY** - no configuration, **CONFIGURED** - CNF has been entered |
| line 5 | **TGT** | DDC and process code (hexadecimal) of subsystem under test |
| line 6 | **#I/F** | number of RID files for the subsystem and Link number 1-8 |
| line 7-12 | **SIV** | all DDCs and process codes (hexadecimal) in the RID files except TGT |
| line 3 | | **-- AUTOTESTER ---** |
| line 4 | **ACTL** name of autotester script or **NONE** | |
| line 5 | **STATE** | **IDLE** - no test is running, **SUSP** - test is suspended, ... |
| line 7 | | **-- FILTERS --** |
| line 8 | **VAL** | - unused |
| line 9 | **RPT** | - unused |
| line 14 | **-- DATA GENERATE --** | |
| line 15 | **STATE** | **E** - RID files in stream table (SLOAD), **D** - no RID files in stream table |
| line 16 | **STATUS** | **OFF** - no data blocks being generated, **ON** - data blocks being generated |
| line 17 | **STREAMS** | number of streams in stream table, 0-6 |
| line 18 | **AG-RATE** | - unused |
| line 19 | **LOGGING** | - unused |
| line 21 | **VALIDATE** | - unused |
| line 22 | **VAL STRM** | - unused |
| line 23 | **LAST BLK** | - unused |
| line 14 | **-- DATA RECEIVE --** | |

| **STS** | SIV Status Display |
|---|---|

| | | |
|---|---|---|
| line 15 | **STATE** | - unused |
| line 16 | **STATUS** | - unused |
| line 17 | **STREAMS** | - unused |
| line 18 | **AG-RATE** | - unused |
| line 19 | **LOGGING** | - unused |
| line 21 | **VALIDATE** | **OFF** - validation off, **ON** - validation on |
| line 22 | **VAL STRM** | name of stream enabled for validation |
| line 23 | **LAST BLK** | last block validated |

# 4

# MESSAGES

## 4.1 Alarms

### 4.1.1 Emergency AlarmsNone

TBS

### 4.1.2 Critical Alarms

None.

### 4.1.3 Warning Alarms

**540:WATCHDOG TIMEOUT FOR TASK ‘aaa’**

The specified task failed to update its watchdog count within the specified period of time. If message 541 does not appear, then the task may have aborted or may be hung.

**541:WATCHDOG - TASK ‘aaa’ STILL ACTIVE**

The watchdog count for the specified task has changed after a timeout was reported. This suggests either the task was unexpectedly delayed or the timeout period is too short.

**700:OFFLINE EN MSGS OCCURRED!! SEE DISPLAY 'ENOFF'**

Before the system was controllable (i.e., while the local terminal is disabled and communication mode is quiescent), one or more event notices were generated. These messages have been routed to a file and can be seen in the display 'ENOFF'.
Bring up display 'ENOFF' and scan the messages.
Take appropriate action(s) based on the messages
Delete the file with 'ENOFF PURGE'.

**796:S/W ERRORS OCCURRED - SEND LOG TO SCOE**

Internal software errors have been detected. There is nothing that the operator can do about these messages. At the end of the pass, the LMC log should be printed and sent to SCOE.

## 4.2 Advisories

### Deviation Advisories

034:TCT HARDWARE DOES NOT EXIST

Output during system initialization if TCT time is not available.

**042:890-131/201 XMIT ERROR(aaa) OF bbb**

Reported when transmission of a generated data block for stream bbb results in error aaa as reported by MSW services.

**099:ERROR IN SRC/DST/MID, 'aaa' NOT CATALOGED**

An error was encountered when attempted to catalog the RID named aaa. The src_code, dst_code, and mid keyword fields should be checked for proper values.

**099:CRIT SET-UP ERROR, VAL ABORT THIS BLK**

A field offset specification in the RID associated with the inbound block was found to be negative with respect to the prior field validated. Validation is aborted for the block.

**099:CSDREG MID=aaa FAIL, ERR=bbb**

The error number bbb was reported by MSW services when attempting to de-register for reception of data blocks with MID aaa (hex) in attempting to process the RCVB directive.

**099:DFLT STREAM NOT SET**

An attempt to remove a stream failed since no steam name was provided and no default stream has been set. (This error should not occur since the default is automatically reset.)

**099:MAX # CATALOG ENTRIES FOR RID (aaa) EXCEEDED**

There are too many RID files within the current directory which is being cataloged as part of the subsystem configuration process (CNF directive). Rename the '.rid' extension of selected files to inhibit read-in or contact software engineering to increase this maximum.

**099:MSW ERR: COULDN'T OPEN DATA FILE aaa**

The file in path aaa does not exist or does not provide read permission.

**099:NO RID TO VAL BLK, SRC=aaa DST=bbb MID=ccc C=ddd**

A block was received which could not be matched to block definition data (SRC, DST, mid) specified in the RIDs which have been cataloged as a result of the CNF directive. This message will be filtered for every 20 blocks received with ddd presenting the tallied block count. Check CNF display for SRC, DST, MID parameters available for match.

**099:RCV STREAM IS GEN STREAM!, ABORTING VALIDATION**

The stream identified for a block received for validation is currently involved in data generation processing. Check the STRM display for stream statuses. Also, verify that routine parameters (SRC, DST, MID) of the received block match those anticipated.

**099:RID LOAD ERROR=aaa ON 'bbb'**

The attempt to load the interface definition file (RID named 'bbb.rid') for validation purposes resulted in error aaa reported by the load process. aaa is one of the following:

**ccc: ln=ddd-->eee.**

An ccc error in line number ddd whose contents are eee was encountered when attempting to load a RID.

**EMPTY FILE--NO FIELDS READ**

RID file attempted for load did not have any fields defined.

**FAILED TO OPEN FILE ccc.**
RID file ccc could not be open. Check RID file directory path and existence of file.
**RAW FILE ccc NOT FOUND FOR ddd**
The raw data file, ccc, specified within the RID for stream ddd could not be located on disk. Raw data files should reside in the operational software directory.
**RID ccc IS ALREADY LOADED**
The stream ccc cannot be loaded since it is already loaded.
**STREAM TABLE FULL (MAX=ccc)**
The maximum number of streams have already been loaded. Use SREM to remove unused streams.
**ccc XREF ERROR=ddd**
An internal software occurred when loading the RID for stream ccc. Record ddd and report it to SIV developers.

**099:STREAM INDEX LOOKUP ERROR FOR 'aaa'**
The RID associated with stream aaa has not been loaded for use during validation. Since RID loading for validation is performed automatically, this indicates a software problem which should be reported.

**099:STREAM NOT LOADED**
An attempt to remove a stream failed since the stream had not been loaded.

**099:STREAM TABLE EMPTY**
An attempt to remove a stream failed since no streams were currently loaded.

**099:TOO MANY SIV HATS FOR TABLE!! MAX=aaa**
The number of unique interfaces (determined by the unique process code) for the target subsystem being configured via the CNF directive exceeds the maximum number currently allowed. Rename the '.rid' extension of the selected files to inhibit read-in or contact software engineering to increase this maximum.

### 4.2.2 Completion Advisories

**043:STREAM aaa ENDED**
Generation of the interface definition stream aaa has terminated.

### 4.2.3 Progress Advisories

**020:BLK DUMPED TO aaa AT bbb**
A data block was written to file aaa at time bbb. If aaa is blank, the block was written to the terminal screen.

**035:TGT PROC/WAIT**
Indicates that a directive has been forwarded to the target (TGT) subsystem and the target has, in turn, responded with a processing/wait directive response.

**040:STREAM aaa STARTED**
Generation of the interface definition stream aaa has begun.

**052:RCVD aaa->bbb ccc CNT=ddd**
A data block of type ccc from source xxx to destination yyy, where xxx and yyy are the DDCs corresponding to the process codes in the data block. ddd data blocks of type ccc were received since the last CNF directive or the last RCVB directive.

### 4.2.4 Log-Only Advisories

**031:ccc: ln=ddd-->eee.**
A ccc error in line ddd whose contents are eee was encountered when attempting to load a RID.

**031:EMPTY FILE--NO FIELDS READ**
RID file attempted for load did not have any fields defined.

**031:FAILED TO OPEN FILE ccc.**

RID file ccc could not be opened. Verify the RID directory path and existence of the file ccc.

**089:MID=aaa OUT OF VALID RANGE (0 TO bbb)**

A block was received which included a message identifier (aaa) which is out of the valid range of values (0 to bbb).

**098:SN aaa COMP bbb LST ccc INIT ddd ENTRIES eee**

An 890-131 FAT data block has been received. The FAT serial number is aaa, the component number is bbb, the last block flag (T=True, F=False) is ccc, the initialize the FAT table flag (T=True, F=False) is ddd, and the number of process codes in this FAT block is eee.

**098:UNPROCESSED MID=aad RECEIVED**

Message output when a block having MID aad is received by has been disabled for normal reporting via the RCVB directive. Debug mode for "RCV" must have been enabled to receive this message.

**797:aaa:bbb:ccc:ddd**

A software error occurred in task aaa where bbb is the source file name, ccc is the source line number, and ddd is the error message. Report this error to the SCOE.

## 4.3 Prompts

None.

# 5

# REPORTS

## 5.1 Reports - Quick Reference

Table 5-1 SIV Reports

| REPORTS | DESCRIPTION | PAGE |
|---|---|---|
| VAL | Validation Report | 5-1 |

## 5.2 Reports - Detailed Descriptions

```
         1         2         3         4         5         6         7         8
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345
159/94                        SIV VALIDATION REPORT                Page: 1

STREAM: con2      CMC (c10) -> TEST (abc)    TYPE: Monitor Data  (19)

-------------------------------------------------------------------------------------

BLOCK #1                         VALIDATED:  159 23:48:22

FIELD NAME    CURRENT VALUE          VALIDATION DEFINITIONS
SEGMENT_ID     127                    CONSTANT: 127
SEGMENT_LEN    35                     CONSTANT: 35
LENGTH         0                      LENGTH: FLDS 0 TO -1
FIELDCK        477                    CHECKSUM: FLDS 0 TO 2
BLOCKCK        0                      CHECKSUM: FLDS 0 TO 3
STRING         'constant'             CONSTANT: 'constant'
INTEGER        -1                     CONSTANT: -1
UNSIGNED       65535                  CONSTANT: 65535
SFLOAT         123.456001             CONSTANT: 123.456000
DFLOAT         987654.321000          CONSTANT: 987654.321000
CHAR           'a'                    CONSTANT: 'a'
MDTIME         '159235459000        ' ASCII TIME
BCD            ******

CENTISEC       8609900                CENTISECONDS
DECISEC        1476158                DECISECONDS
BINDOY         159                    DAY-OF-YR: 1 TO 365
BCDDOY         0                      BCD DAY-OF-YR: 1 TO 365
```

Figure 5-1  Validation Report

DESCRIPTION

The SIV Validation Report provides for review of the content of each block received in a readable format. Specifically, the report includes two components:

- Stream information identifying the source, destination, and type of data
- formatted data block contents including:
  - block count
  - validation time
  - current values matched to field names
  - validation definitions extracted from RID

NOTES

Validation errors are flagged using the text:

"***VALERR #n *** <error message>"

This message is output directly below the field item causing the error.

The VAL directive identifies which inbound data stream will be validated with a corresponding report.

Use the STS display to determine which, if any, stream has validation enabled.

LIMITATIONS

Currently, only setup errors that inhibit data value extraction are detected and reported. Further validation of values must manually be performed until SIV Build 2.

Certain data types defined within the RID are not yet handled in validation. Specifically, BCD, Modcomp float, and Fixed point data types are not yet supported.

Only one validation report can be generated at a given time.

MESSAGES

**Invalid #bits, INT type #bits range is 1 to 64**
(Range should state 1 to 32). This message will be written to the report whenever the number of bits defined in the RID exceeds 32 bits for the 'Integer data type. Value conversion will not be performed.

**Invalid #bits, UNSIGNED type #bits range is 1 to 64**
(Range should state 1 to 32). This message will be written to the report whenever the number of bits defined in the RID exceeds 32 bits for the 'Unsigned data type. Value conversion will not be performed.

**Invalid #bits, FLOAT type #bits range is 1 to 64**

This message will be written to the report whenever the number of bits defined in the RID exceeds 64 bits for the 'Float data type. Value conversion will not be performed.

**MODCOMP / FIXED POINT float types are not supported (YET)**
This message will be written to the report whenever the 'M' or 'Fn' data types are encountered within a RID definition. Value conversion will not be performed.

**Unrecognized data type in RID definition for this field**
This message will be written to the report whenever the RID data type is not 'B', 'C', 'F', 'Fn', 'I', 'M', 'T', or 'U'. Value conversion will not be performed.

**Character field overruns defined maximum (40)**
This message will be written to the report whenever the number of bits defined for a 'C' data type exceeds 320 (i.e., 40 8-bit chars). Value will be truncated to 40 characters.

| | | |
|---|---|---|
| Line 1: | **doy/yr** | day of year and 2-digit year |
| | **title** | 'SIV VALIDATION REPORT |
| | **page: n** | page number |

| | | | |
|---|---|---|---|
| Line 3: | **stream id** | - | stream identifier matching RID filename |
| | **source** | | Source DDC and process code (hex) |
| | **dest** | | Destination DDC and process code (hex) |
| | **type** | | Data block description and message id (hex) |

BLOCK DATA - repeats with each new block received

Line 5: solid line separating block information from prior block or from page header

| | | | |
|---|---|---|---|
| Line 7: | **block #** | | number to identify blocks received for validation |
| | **val time** | | day of year and time block received for validation |
| Lines 9-n: | **field id** | | field name from RID definition (or FLDnnn if none defined) |
| | **value** | | current value extracted from received data block |
| | **RID defn** | - | validation definitions for data type and valid range (when applicable) |
| | **CONSTANT:** | | constant value expected |
| | **LENGTH**: | | block length determined by field count |
| | **CHECKSUM:** | | block segment checksum based on field range |
| | **ASCII TIME:** | | Monitor Data Time field defined in 890-132 |
| | **IN RANGE:** | | Value range for numeric data types |
| | **FROM SET:** | | Discrete set of values listed |
| | **MILLISECS:** | | Numeric value will be time in milliseconds |
| | **CENTISECS:** | | Numeric value will be time in centiseconds |
| | **DECISECS:** | | Numeric value will be time in deciseconds |
| | **DAY-OF-YR:** | | Numeric value will be day of year |
| | **BCD DAY-OF-YR:** | | Numeric value will be BCD day of year |
| | **UNKNOWN ACTN:** | | Action value presented in char and hex since it was not expected for this data type |

**THIS PAGE INTENTIONALLY LEFT BLANK**

# 6

# RAPID INTERFACE DEFINITION (RID)

SIV uses the Rapid Interface Definition (RID) to define a subsystem's interface. RID files (RIDs) are translations of Subsystem Interface Agreements (SIA) or more formally, "820-16 ; Deep Space Network System Requirements - Detailed Subsystem Interface Design". This chapter describes the RID.

Section 6.1 defines the lexical components and syntax of the RID file. Section 6.2 provides a step-by-step guide on how to create a RID given an interface agreement. Section 6.3 describes the tools used to create a RID. Section 6.4 describes the changes to the RID definition in this release of SIV. Section 6.5 describes supplanted, but still supported, backward-compatible features.

## 6.1 RID ASCII File Format

SIV RIDs are the means to describe to the SIV the way the subsystem wants it to generate data blocks. The SIA is the basis for creating RIDs. SIV cannot read the SIA directly, because the SIA defines a range of values, while a test definition may need a subset of that range or values outside the range.

There is an SIA-to-RID translator (xlate -- see section 6.3.4) that creates a base RID from a SIA conforming to the 1995 SIA standard (see Section 6.3.5). Human experts must still create Test RIDs, because the expert can determine meaningful test sets while the translator can only determine legal or inclusive ranges. The Test RIDs are then created from the base RID file and not the SIA.

A base RID can be created manually. The most common practice has been to copy a RID from another subsystem and then edit it in any text editor.

The RID files have the naming convention "<name>.rid", where name has a 8 character maximum length, it must follow the normal naming conventions of the disk operating system, and all the letters must be *lower-case*.

A RID file is an ASCII file composed of *Transmission records* and *Interface records*. The Transmission records describe the stream transmission, including protocol parameters, block transmission frequency and count. Interface records define the data block or segment format. A data block is a set of data fields. Data fields are defined by *field records* or the contents of a *raw data* file.

Table 6-1 Rapid Interface Definition File Layout

| |
|---|
| Transmission Records<br>- Block Transmission Controls: COM<br>- 890-131: MID, LAN Selection, Protocol, SAC<br>- 890-201: data type, block type |
| Interface Records<br>-Field Records<br>-Raw Data Records |

Table 6-2 Transmission Record Format

| keyword | value | [comment] |
|---|---|---|

Table 6-3  Field Record Format

| | | | | | |
|---|---|---|---|---|---|
| field name | length[@offset] | data type | action | [value […]] | [comment] |

Table 6-4  Raw Data Record Format

| | | | | |
|---|---|---|---|---|
| keyword | filename | segment size | [word fill] | [comment] |

Table 6-5  Value Representation Formats

| | |
|---|---|
| Integer or Scalar | Integer representations follow the standard "C" representations.  A hexadecimal value has a "0x" or "0X" prefix.  An octal value has a "0" prefix.  A decimal value has no prefix.  Commas, or any punctuation (except an optional leading "+" or "-" sign in a signed integer), are not permissible in integers (for example 10,000). |
| Floating Point | Floating point representations follow the IEEE standard, using digits, a decimal point, the letter "e" or "E", and "+" or "-" sign leading the number or the letter "e".  Commas are not permissible.  For example, "11", "-1.1", "+.1", "1.1e2", "1e-2", and "0E+2", are all legitimate -- while "10,000.01" is not. |
| Text or String | Text words or strings are white-space delimited sequences of characters.  A text word may be a quoted string, which can contain space characters.  Quoted strings use the double quote characters: "...", but the quotes are not word delimiters.  The double quote character may be included in a text word by escaping the quote: \".  Spaces may also be escaped, for example, 'c\ ab' is equivalent "c ab".  * |

*   The single quote character is used here only for illustration and has no special meaning in RID.
Unless otherwise noted, keywords, field names, data types, actions, and other forms of text entry are not case-sensitive.

### 6.1.1 Transmission Record Description

Transmission records specify the routing and transmission parameters. Default values exist for most keyword parameters. Keywords without a default value are required and must be included in a RID file.

General Syntax: <keyword> [<value>] [#<comment>]
In the tables, under *Value Range*, "<value>" gives the value *type* and "{ ... }", "[ ... ]", or "( ... )" gives the value *range* ("[]" denote inclusive and "()" denote exclusive range limits, while "{}" denote sets). A RID value outside the range is an error.

Keywords marked as I(nput), under 131 or 201, are used by SIV to match incoming data blocks with the appropriate logging and validation RID file. Keywords marked as O(utput) under, 131 or 201, are used by SIV to generate outbound data blocks.

Table 6-6 Transmission Record Keywords

| Keyword | Description | Value Range | Default Value |
|---|---|---|---|
| comm_protocol | Communication Protocol | <text> = { 131, 201 } | *mid* < 10 ? 201 : 301 |
| data_begin | Begin data section (and end header sect.) | no value argument | N/A |
| data_end | End data section (and begin trailer sect.) | no value argument | N/A |
| ssend_count | Data block or segment count per SSEND | <integer> = [ 0, 32767 ] * | 1 |
| ssend_delay | Time in seconds btwn. blocks per SSEND | <float> = [ 0.1, $10^6$ ) | 1.0 |

Table 6-6 Transmission Record Keywords (Continued)

| | **890-131 & 890-201 Protocols Keywords** | | | **131** | **201** |
|---|---|---|---|---|---|
| block_type | DFL-1-1 Block Type | <text> = { DDD, NON, OPS_6_7, OPS_6_8, ... } ** | DDD | | O |
| mid | Message Id | <integer> = 201:[ 0, 9 ], 131:[ 0xA, 0xFF ] | *required* | I/O | I/O |
| proc_code_des | Destination subsystem's | <integer> = [ 0, 0xFFF ] | *required* | I/O | I/O |
| proc_code_src | Source subsystem's Process Code | <integer> = [ 0, 0xFFF ] | *required* | I/O | I/O |
| sac | Special Application Code | <integer> = [ 0, 0xFF ] ** | 0 | O | O |
| sdb_data_type | Standard Data Block (SDB) data_type | <integer> = [ 0, 0x7F ] | 0 | | I |
| select_lan | LAN Selection | <text> = { XSANYLAN, XSHRLAN, XSSPCLAN } | XSANYLAN | O | |
| select_protocol | Protocol Selection | <text> = { XSDP, XSNAP, XSPAR } | XSNAP | O | |
| sub_block_id | 131 MDTS Id or 201 NOCC R/T Packet Id | <integer> = [ 0, 0x7F ] | 0 | I | I |

* A zero value tells SIV to transmit this block continuously and can be regarded as equivalent to infinity.

** There are more, but much less common in SIV, *block_type* values. See the *Keyword Description* for *block_type* below.

*** The value range for an 890-201 *sac* is different from the 890-131 range. Some 201 *sac* values can turn off transmission!

#### 6.1.1.1 Transmission Record Keyword Descriptions

The keyword value ranges are listed in the **Error! Reference source not found.**. The ranges are not duplicated in the following descriptions unless there is an accompanying explanation for the values. Ranges listed as "<integer> =" may have decimal, octal, or hexadecimal record values.

##### 6.1.1.1.1 General Keywords

**comm_protocol** This specifies the Communication Protocol used transmit the block:

| | |
|---|---|
| **131** | Use 890-131 to transmit the data block. |
| **201** | Use 890-201 to transmit the data block. |

If *comm_protocol* is not present in the RID file, then the *comm_protocol* will be automatically set to 201 if *mid* is less than 10, otherwise it will be set to 131. ( *comm_protocol* = *mid* < 10 ? 201 : 131 )

**data_begin, data_end** This identifies the beginning and of the data portion, excluding the header and the trailer. This is used with the validation feature of the SIV. This keyword has no effect unless logging - *without headers* - is enabled (see the LOG and VAL directives - validation validates a log file). This keyword tells SIV to skip the data fields preceding *data_begin* and following *data_end* in the RID file when validating a log file because the incoming data block was logged without the header or the trailer.

These keywords are designed for writing RID files capable of validating the protocol header and trailer, and allowing this feature to be turned on or off dynamically. These keywords have no effect on outbound block generation.

The 890-131 headers are never logged. The 890-201 headers and trailers are only logged if explicitly requested.

**ssend_count** This is the number of data blocks to transmit per SSEND directive, except when using raw data records. When using raw data records, the value of *ssend_count* represents the number of transmission cycles per SSEND. The number of blocks it takes to read the entire raw data file is one transmission cycle.

A zero value tells SIV to transmit this block continuously and can be regarded as equivalent to infinity.

**ssend_delay** This is the number of seconds between each transmission when transmitting multiple data blocks with one SSEND directive. Accurate to the tenth of a second.

#### 6.1.1.1.2 890-131 & 890-201 Protocols Keywords

**block_type** This is the DFL-1-1 (890-201) Block Type. This keyword is meaningful only when

*comm_protocol* = 201.

The most common SIV values for the *block_type* are:

**DDD** Standard Data Block (SDB) format and not an encapsulated OPS-6-7/8. (The block must already be in SDB format in the RID file.) This is the default value.

**NON** Non-SDB block: this is for application blocks, but not data, in their own format.

**OPS_6_7** OPS-6-7 block: Goddard format block to be encapsulated in 890-201.

**OPS_6_8** OPS-6-8 block: old DSN format block to be encapsulated in 890-201.

Plus the following less common:

**IF_INIT** IF INIT block: this is a 131 interface initialization of the DEST PARAM. *

**INIT** INIT block: this is a 201 initialization for table set-up and reading router files. *

**PRESET** This is an already encapsulated 201 block.

* SIV automatically sends the necessary 890-201 initialization blocks.

And for completeness:

**CLOSE**, **DFMC**, **IP**, **RCB**, **REGISTER**, **STATUS**, **TERMINATE**, **WIN**.

See the 201 User's Guide [MISSION-DEPENDENT EQUIPMENT (MDE) DSN NETWORK-LEVEL DATA-FLOW COMMON SOFTWARE (890-201)] for a more complete definition of these block types.

**mid** This is the 890-131 Message Id and is a *required* keyword. This is one of the keywords that uniquely identify the RID file, matching it with incoming data blocks, for logging and validation.

**proc_code_dest** This is the 890-131 Process Code of the destination subsystem that this data is *sent to* and is a *required* keyword. This is one of the keywords that uniquely identify the RID file, matching it with incoming data blocks, for logging and validation.

**proc_code_src** This is the 890-131 Process Code of the source subsystem that this data is *received from* and is a *required* keyword. This is one of the keywords that uniquely identify the RID file, matching it with incoming data blocks, for logging and validation.

**sac** This is the 890-131 Special Application Code. This is one of the keywords that uniquely identify the RID file, matching it with incoming data blocks, for both logging and validation.
Be careful when *comm_protocol* = 201. The value range for an 890-201 *sac* is not only different from the 890-131 range, but some 890-201 *sac* values will turn off the transmission!

**sdb_data_type** This is the 890-201 standard data block (SDB) data_type. (This is the 201 SDB header parameter *data_type*.) This is one of the keywords that uniquely identify the RID file, matching it with incoming data blocks, for logging and validation. This keyword is not used to generate outbound data blocks.

**select_lan** This is the 890-131 LAN Select parameter. The available LAN types are:

**XSANYLAN** Use any LAN available. This is the default value.

**XSHRLAN** Use the High Rate LAN.

**XSSPCLAN** Use the SPC LAN.

**select_protocol** This is the 890-131 Protocol Select parameter. The available protocols are:

**XSDP** Transmit using the direct protocol.

**XSNAP** Use No Acknowledgment Protocol to transmit. This is the default value.

**XSPAR** Use Positive Acknowledgment Received to transmit.

**sub_block_id** This is either the 890-132 Monitor Data Transfer Segment (MDTS) segment identification or the 890-201 NOCC real-time (R/T) packet identifier. This is one of the keywords that uniquely identify the RID file, matching it with incoming data blocks, for logging and validation. This keyword is not used to generate outbound data blocks.
The *sub_block_id* is the MDTS segment identification when *comm_protocol* = 131 and *mid* = 0x19. The *sub_block_id* is the R/T packet identifier when *comm_protocol* = 201, *mid* = 0x05, *block_type* = DDD, and *sdb_data_type* = 0x41. The *sub_block_id* is not meaningful under any other circumstances.

### 6.1.2 Field Record Description

Field records define each data item within a data block. The general limitations for the field records are: 1 record per line, 255 characters per record, and 500 records per RID file.

The 500 record limit may be extended to 2999 records, by loading (SLOAD) the 500+ record RID by itself and removing (SREM) it before loading any other RIDs. SIV will enforce this restriction of only one 500+ record stream at a time.

The record upper limit includes the segment size of the raw data. See *Record Indices* in section 6.1.3 - Raw Data Record Description.

General Syntax: <name> <length>[@<offset>] <type> <action> [<value> …]

where:

**name** The *name* for the field record is a text word of 2 to 13 characters. The *name* must start with a letter, be unique within the RID file, and cannot be a keyword name.

**length** The *length* (size) of the data item is in bytes and bits and specified by "[<bytes>:][<bits>]". The *length* is the *sum* of the byte and bit values. The maximum data *length* is 32 bytes or 256 bits. See the *offset* section for an example. There may be additional length restrictions associated with each data type and action.

**offset** The optional *offset* position of the data is in bytes and bits and specified by "<*length*>@[<bytes>:][<bits>]". The *offset* is the *sum* of the byte and bit values. The *offset* is an absolute position from the beginning of the block (starting at bit 0) and is not relative to the field record's position.

As an example, the following will position a field at word 296 (16 bit words) within the block. Bit 4720 is the starting bit for word 296 (16 x (296 - 1) = 4720). Given a data item 16 bits long, 2:0@590:0, 2:@590:, 16@590:, 16@4720, or 2:@4720 are equally correct in defining the length and start location of the value.
Without the optional *offset*, the location of a data record will directly following the bit position of the previous data record. This also applies to raw data and data records that follow the raw data. See *Record Indices* in section 6.1.3 - Raw Data Record Description.

It is an error if the offset is less than the calculated bit position for that field record. If the offset is greater than the calculated bit position for that field record, the field record value will be copied to the offset position, with the bits between the preceding field value and the offset set to zero, and all subsequent field record values will follow this new bit position. In each use of the offset, either a diagnostic or an error message will be issued.

**type** The data type for the field of data being described:
In the table, "[ ... ]" and "( ... )" gives the values' maximum *range*, determined by the SIV implementation bit size limits ("[]" denote inclusive and "()" denote exclusive range limits). Fields defined smaller than the the maximum range's bit size, will have smaller ranges. A RID value outside the range is an error.

Table 6-7 Field Record Data Types

| Type | Description | **Length** | **Value Range** |
|---|---|---|---|
| B | Binary Coded Decimal (BCD) | * | value ≤ 18 digits ( $-10^{18}$, $10^{18}$ ) ** *** |
| C | Character or string of characters | * | each character in a value requires 8 bits, for a 32 byte or 256 bit maximum |
| F | IEEE floating point | 32, 64 | IEEE floating point single and double precision limits |
| F<n> | Fixed Point Scaled Integer with *n* fractional bits | * | value ≤ 32 + *n* bits with *n* ≤ 32 ( -2147483648.0, 2147483648.0 ) *** **** |
| I | Signed Integer | * | value ≤ 32 bits [ -2147483648, 2147483647 ] **** |
| M | Modcomp II floating point | 32, 48 | Modcomp II floating point single and double precision limits |
| T | Monitor Data Time | 96 | N/A |
| U | Unsigned Integer | * | value ≤ 32 bits [ 0, 4294967295 ] |

* Data Types without field lengths are only limited by the maximum value of data length (32 bytes or 256 bits).

** SIV does not implement floating point BCDs.

*** Very large values will suffer from least significant bit loss due to SIV implementation limits (IEEE double precision float).

**** Negative values of any size cannot be represented in fields larger than 32 (+*n*) bits.

Section 6.1.2.1 provides a detailed discussion of Data Types, including the algorithms that determine the ranges.

**action** The algorithm to perform when generating the data value.

Table 6-8 Field Record Actions

| Action | Description | Supported Data Types | | | | | | | | Length | VAL ** | Values |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **B** | **C** | **F** | **Fn** | **I** | **M** | **T** | **U** | | ** | |
| + | Increment from initial value | • | | • | • | • | • | | • | * | [ ] | <low> <high> <step> |
| - | Decrement from initial value | • | | • | • | • | • | | • | * | [ ] | <low> <high> <step> |
| C | Constant value | • | • | • | • | • | • | | • | * | = | <constant> |
| P | Pick randomly from list of values | • | • | • | • | • | • | | • | * | = | <value-1> [ <value-2> … <value-16> ] |
| R | Random value between two values | • | | • | • | • | • | | • | * | [ ] | <low> <high> |
| S | Sequence through list of values | • | • | • | • | • | • | | • | * | = | <value-1> [ <value-2> … <value-16> ] |
| V | Variable action based on SLOAD | • | | • | • | • | • | | • | * | • | <low> <high> <step> |
| | **Data Block Actions** | | | | | | | | | | | |
| F | Between Bytes Block Checksum | | | | | | | | • | 0 **** | | <start-field > <end-field > <location> <last-byte> *** |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K | Checksum | | | | | | | | • | 16 **** | | <start-field > <end-field > *** |
| L | Byte length between field records | | | | | | | | • | * | = | <start-field > <end-field > *** |

**Raw Data Actions**

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| @ | Block Sequence Number | • | | • | • | • | • | | • | * | [ | <low> <high> <step> |
| A | Last Block Boolean Algebra | • | | • | • | • | • | | • | * | = | <true-value> <false-value> |
| Z | Raw Data Size Calculator | | | | | | | | • | 16 | [ | <offset> |

**Time and Date Actions**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | No action | | | | | | | • | | 96 | |
| D | Day of Year (DOY) in BCD format | | | | | | | | • | 10 | [ ] |
| H | Time of day in milliseconds | | | | | | | | • | 27 | |
| M | Time of day in deciseconds | | | | | | | | • | 24 | |
| T | Time of day in centiseconds | | | | | | | | • | 24 | |
| Y | Day of Year (DOY) | | | | | | | | • | 16 | [ ] |

* Actions without a field length have no length limits. The field lengths are only limited by the data type.

** Validation symbols: = test for equality with value(s), [ test for ≥ low value, ] test for ≤ high value.

*** Values that specify other field records may be field indices, field names, or the letters {P, N, E} (previous, next, end).

**** These actions require 16-bit word aligned fields and record values.

Section 6.1.2.2 provides a detailed discussion of Actions.

**value** … These are the RID values used by the action to determine the contents (field value) of this field when the block is generated. The number of values is dependent on the action for the field, but cannot exceed 16.

A *record value* is any value following an action in a RID file or set with the FVAL and FACT directives.

A *field value* is the value generated by SIV for that field for transmission to a subsystem.

A data type, an action, and the action's *record values* determine the generated *field values*.
An integer record value's format (number base) determines the format in the various SIV outputs, such as the Validation report. If the values are hexadecimal, octal, or decimal, the format of the outputs will be the same (see Table 5 - Value Representation Formats).

Values that specify other field records may be field indices, field names, or the letters {P, N, E} (for previous, next, and end field, respectively). A field's index is its order within the RID file starting with 0 and ending at field N - 1.

### 6.1.2.1 Data Type Descriptions

The following describes the data types used to define field items. Each data type may have both field length and value range limitations. If no limitations are listed, then the length limitation is the same as the maximum value for the field length (32 bytes or 256 bits) and the value limitation is the largest value that can be represented in any field length. The value ranges use the notation "[]" and "()" which denote inclusive and exclusive range limits, respectively.

## B Binary Coded Decimal Type

This type formats the data as a Binary Coded Decimal (BCD).

A BCD has each digit of a decimal number stored in 1 to 4 bits. For example, since { 0 = 0, 1 = 1, 2 = 10, …, 9 = 1001 }, then 99 is stored in 8 bits as 10011001 and the Day of Year (DOY) can be stored in 10 bits.

The range of values is ( $-2^{(m \% 4)} \times 10^{(m / 4)}$, $2^{(m \% 4)} \times 10^{(m / 4)}$ ), where *m* is the bit length of the field and % is the division remainder operator. The maximum value supported by the SIV is limited by 18 decimal digits. This does not limit the number of bits that can be specified -- but values values outside the bounds of ( $-10^{18}$, $10^{18}$ ) cannot be represented.

Floating point BCDs are not implemented in SIV.

Very large values will suffer from least significant bit loss due to SIV implementation limits (IEEE double precision float).

Record values must be decimal (see Table 5 - Value Representation Formats).

## C Character Type

This type formats the data as a Character or a string of Characters. Embedded spaces can be defined through use of double quotes as in: "My Data".

The Character type must have at least 8 bits for the field to represent a valid character. Fields with defined lengths that are not exactly divisible by 8 will have the trailing 1 to 7 bits set to zero.

String values with lengths greater than the field length will be truncated with a warning. String values with lengths less the field length will be silently left-justified and padded with spaces. The character string will not be NUL (zero) terminated.

The maximum string size is 32 bytes.

## F IEEE Floating Point Type

This type formats the data as an IEEE Floating Point number.

The value and precision limits are the IEEE single and double precision limits.

The field length must be 32 bits for single-precision or 64 bits for double-precision values.
Record values must be floating point (see Table 5 - Value Representation Formats).

## F<n> Fixed Point Scaled Integer Type

This type formats the data as a Fixed Point Scaled Integer (FPSI) with *n* fractional bits, where *n* must be within [ 1, 31 ].

A FPSI is a *m* bit two's complement signed integer converted to floating point number by dividing it by $2^n$, so the whole number portion of the word is the upper $m - n$ bits. This means the largest number for a FPSI is $(2^{(m-1)} - 1) / 2^n$, the smallest fraction is $1/2^n$, and the only representable fractions are multiples of $1/2^n$.

The range of values is $[ -2^{(m-1-n)}, 2^{(m-1-n)} - 1/2^n ]$, where *m* is the bit length of the field. The maximum value supported by the SIV is limited by $m \leq 32 + n$ bits. This does not limit the number of bits that can be specified -- but values outside the bounds of $[ -2147483648, 2147483648 - 1/2^n ]$ and absolute non-zero values less than $| 4.6566 \times 10^{-10} | (2^{-31})$ cannot be represented.

Fractional record values that are not exact multiples of $1/2^n$ will produce *truncated* field values of the largest multiple of $1/2^n$ that is less than the record value. e.g. A F2 record value of 1.9999 yields a data value of 1.75 and a record value of 1.1 yields 1.0.

Very large values will suffer from least significant bit loss due to SIV implementation limits (IEEE double precision float).

Negative values of any size cannot be represented in fields larger than 32 + *n* bits.

Record values must be floating point (see Table 5 - Value Representation Formats).

## I Signed Integer Type

This type formats the data as a Signed Integer using two's complement format.

The range of values is [ $-2^{(m-1)}$, $2^{(m-1)} - 1$ ], where *m* is the bit length of the field. The maximum value that is supported by the SIV is limited by 32 bits. This does not limit the number of bits that can be specified -- but values but values outside the bounds of [ -2147483648, 2147483647 ] cannot be represented.

If more than 32 bits are necessary to specify a non-zero value, a possible solution is to define adjacent fields of the **U** type.

Negative values of any size cannot be represented in fields larger than 32 bits.

Record values may be decimal, octal, or hexadecimal (see Table 5 - Value Representation Formats).

**M Modcomp II Floating Point Type**

This type formats the data as a Modcomp II floating Point number.

The value and precision limits are the Modcomp II single and double precision limits.

The field length must be 32 bits for single-precision or 48 bits for double-precision values.

Record values must be floating point (see Table 5 - Value Representation Formats).

**T Monitor Data Time Type**

This type formats the data as a Monitor Data Time (Character) string. The field length must be 96 bits. This type has no acti on or values. The type will retrieve the system/TCT time value when the stream is generated and format it into the Monitor Data Time string as defined by 890-132.

**U Unsigned Integer Type**

This type formats the data as an Unsigned Integer, also known as hexadecimal format.

The range of values is [ 0, $2^m$ - 1 ], where *m* is the bit length of the field. The maximum value that is supported by the SIV is limited by 32 bits. This does not limit the number of bits that can be specified -- but values outside the bounds of [ 0, 4294967295 ] cannot be represented.

When one subsystem data field contains multiple bit fields, it can simplify testing to break the subsystem field into multiple RID records, one for each bit field.

If more than 32 bits are necessary to specify a non-zero value, a possible solution may be to define adjacent fields of 32 bits or less and set their values as related constants (**C** action) or sequences (**S** action).

Record values may be decimal, octal, or hexadecimal (see Table 5 - Value Representation Formats).

### 6.1.2.2 Action Descriptions

Actions are the algorithms used to create the field's value from the record's values. When manipulating data through the use of the FVAL and FACT directives within autotester scripts the most common action, with few exceptions, will be the Constant (**C**) action.

#### 6.1.2.2.1 General Actions

**+ Increment Action**

This action increments the field value, starting with the *low* record value, ending with the *high* record value and incrementing by the *step* record value. When the high value is reached, the SIV will wrap around, starting over again with the low value as the initial starting point.

Examples:

| # | **values** | | | | | | |
|---|---|---|---|---|---|---|---|
| # **name** | **length** | **type** | **action** | *low* | *high* | *step* | **comments** |
| GEN_COUNT | 16 | I | + | 0 | 10 | 1 | # Increment example |

This action requires three record values:  1) a *low* value  2) a *high* value and  3) a *step* value.  In the example, the *low* value is 0, the *high* value is 10 and the *step* value is 1.  The first 11 generations of the field will have values of 0 through 10.  The 12th generation will wrap around to 0 and begin again.  After each generation, the field value will be incremented by 1.

Limitations:

This action is compatible with the **B**, **F**, **Fn**, **I**, **M** and **U** data types.  This action has no field length limitations.

**- Decrement Action**

This action decrements the field value, starting with the *high* record value, ending with the *low* record value and decrementing by the *step* record value.  When the *low* value is reached, the SIV will wrap around, starting over again with the *high* value as the initial starting point.  Example field record:

Examples:

| # | **values** | | | | | | |
|---|---|---|---|---|---|---|---|
| # **name** | **length** | **type** | **action** | **low** | **high** | **step** | **comments** |
| T_MINUS | 16 | I | - | 0 | 10 | 1 | # Decrement example |

This action requires three record values:  1) a *low* value  2) a *high* value and  3) a *step* value.  In the example, the *low* value is 0, the *high* value is 10 and the *step* value is 1.  The first 11 generations of the field will have values of 10 through 0.  The 12th generation will cause the SIV to wrap around to 10 and begin again.  After each generation, the field will be decremented by 1.

Limitations:

This action is compatible with the **B**, **F**, **Fn**, **I**, **M** and **U** data types. This action has no field length limitations.

**C  Constant Action**

This action will set the field value to the record value in every block transmitted.

Examples:

| # name | length | type | action | values | comments |
|---|---|---|---|---|---|
| SDSET | 64 | C | C | NSS__DGT | # 20 Set Name (subsystem defined) |

In this example, NSS__DGT will be the field record's value throughout the transmission cycle.

Limitations:

This action is compatible with the **B**, **C**, **F**, **Fn**, **I**, **M** and **U** data types. This action has no field length limitations.

**P  Pick Action**

This action determines the data value by randomly choosing one of the record values. There must be at least 1 value (though only 1 value is equivalent to the Constant (**C**) action), but no more than 16 values.

Examples:

| # name | length | type | action | values | comments |
|---|---|---|---|---|---|
| PICK_DAT | 32 | U | P | 10 67 32 | # Pick example |

At generation time, the SIV will randomly pick one of these three numbers to fill the field with.

Limitations:

This action is compatible with the **B**, **C**, **F**, **Fn**, **I**, **M** and **U** data types. This action has no field length limitations.

**R Random Action**

This action generates random data values within the range of the *low* and *high* record values.

Examples:

| # name | length | type | action | values | comments |
|---|---|---|---|---|---|
| RAND_DAT | 32 | U | R | 10 32767 | # Random example |

In the above example, random numbers between 10 and 32767 will be generated by the SIV at stream generation time.

Limitations:

This action is compatible with the **B**, **F**, **Fn**, **I**, **M** and **U** data types. This action has no field length limitations.

**S Sequence Action**

This action determines the data value by sequencing through the record values, in the given order. There must be at least 1 value (though only 1 value is equivalent to the Constant (**C**) action), but no more than 16 values.

Examples:

| # name | length | type | action | values | comments |
|---|---|---|---|---|---|
| SEQ_DAT | 64 | C | S | DSN NETWORK USA | # Sequence example |

In the above example, SIV will generate the field the value DSN on the first generation, "NETWORK" on the second generation, "USA" on the third generation, "DSN", again, on the fourth generation and continue the cycle with each subsequent generation.

Limitations:

This action is compatible with the **B**, **C**, **F**, **Fn**, **I**, **M** and **U** data types. This action has no field length limitations.

**V Variable Action**

This action has no unique algorithm, and only uses other actions' algorithms. This defaults to the Random (**R**) action, *but can be changed dynamically* to use either the Increment (**+**), Decrement (**-**), or Constant (**C**) actions.

If a Variable action parameter is not given with the SLOAD directive when loading the RID, the Variable action will behave exactly like the Random action (ignoring the *step* record value.) If a Variable action parameter is given with the SLOAD directive, *every Variable action in the RID file will be changed!*

A Variable action requires three record values: 1) A *low* value 2) A *high* value 3) and a *step* value.

The Variable action parameters to SLOAD are:

**+AS** Every Variable action will change to the Increment (+) action with the *low* record value being the initial starting value, the *high* record value being the ending value and the *step* record value being the incrementing value.

**+DE** Every Variable action will change to the Decrement (**-**) action with the *high* record value being the initial starting value, the *low* record value being the ending value and the *step* record value being the decrementing value.

**+LO** Every Variable action will change to a Constant (**C**) action with its assigned value being the *low* record value.

**+HI** Every Variable action will change to a Constant (**C**) action with its assigned value being the *high* record value.

**+MI** Every Variable action will change to a Constant (**C**) action with its assigned value being a SIV determined mid-range value between the *low* and *high* record values { value = ( *low* + *high* ) / 2 }.

Examples:

```
# Filename: stream22.rid
#…
#                    values
```

```
# name length  type  action  low  high  step  comments
VARDAT   16    I     V        1   10    1     # Variable example
# … EOF
```

```
SIV TERMINAL>> SLOAD  stream22        # use the Random action on VARDAT
SIV TERMINAL>> SLOAD stream22 +DE     # use the Decrement action on VARDAT
SIV TERMINAL>> SLOAD stream22 +HI     # use high for the Constant action on VARDAT
```

Limitations:

This action is compatible with the **B**, **F**, **Fn**, **I**, **M** and **U** data types. This action has no field length limitations.

#### 6.1.2.2.2 Data Block Actions

Values that specify other field records may be field indices, field names, or the letters {P, N, E} (for previous, next, and end field, respectively). A field's index is its order within the RID file starting with 0 and ending at field N - 1.

**F Between Bytes Block Checksum Action**

This action will compute a block checksum between the two specified field records: *start-field* and *end-field*, and copy the checksum to the byte offset *location*.

The checksum calculation will end at the byte offset of *last-byte*. (The reasoning behind the *last-byte* record value is that during development the block may not have meaningful values between *last-byte* and the end of the block at *end-field*.)

Examples:

| # | | | | values | _ | | | |
|---|---|---|---|---|---|---|---|---|
| **# name** | **length** | **type** | **action** | **start-field** | **end-field** | **location** | **last-byte** | **comments** |
| BLKCKSUM | 0 | U | F | SOMEFIELD | P | 160 | 159 | # Block Checksum |

Limitations:

This action requires the **U** data type and a field length of 0 bits. The locations specified by the record values must be aligned on 16-bit word boundaries. There can only be one **F** action per RID file. The checksum data record must follow (though it doesn't matter where) the *start-field* and *end-field* data records in the RID file.

Warnings:

The record values *location* and *last-byte* are absolute byte offsets from the beginning of the block specified by the RID (field record index zero) and are not relative offsets from *start-field*.

Be very careful when determining the byte offset *location*, as the 16-bit *location* will be overwritten with the checksum value.

The byte offset *last-byte* must be within the bounds of *start-field* and *end-field*.

The two checksum actions **F** and **K** have different checksum algorithms for the checksum value (see Notes).

Notes:

The checksum is an exclusive-or (XOR) of the 16-bit words in the specified memory block. Prior to the XOR of each word with the checksum, the 16 checksum bits are shifted one bit toward the high bits and the old high bit is copied to the low bit.

## K_ Checksum Action

This action will compute a checksum inclusively between the two specified field records: *start-field* and *end-field*.

Examples:

| # **keyword** | **raw data file** | **segment size** | **word fill value** | **comment** |
|---|---|---|---|---|
| rawdat tlm3204.raw | 271 | 0x1000 | | # see section 6.1.3 |

| # | | | **values** | _ | | |
|---|---|---|---|---|---|---|
| # **name** | **length** | **type** | **action** | **start-field** | **end-field** | **comments** |
| SDCKSUM | 16 | U | K | SDSDLN | P | # Checksum example |

The above checksum field record, references field SDSDLN (field 13) and *P* - the previous field (which equates to field 295. Because the example includes raw data, the raw data makes up the 271 additional records, where 271 is the rawdat segment size.)

Limitations:

This action requires the **U** data type, a field length of 16 bits, and must be aligned on a word boundary. The locations specified by the record values must be aligned on 16-bit word boundaries. The checksum data record must follow (though it doesn't matter where) the *start-field* and *end-field* data records in the RID file.

Warnings:

The two checksum actions **F** and **K** have different checksum algorithms for the checksum value (see Notes).

Notes:

The checksum is the two's compliment of the sum of all 16 bit words inclusively between the two specified field records.

**L Data Length Action**

This action is used for fields which specify a byte length count such as the "length" field of a CHDO header. The field value is set to the byte count of all the fields inclusively between the two record values.

Examples:

```
#                         ___values___
# name length  type   action  start-field end-field   comments
CHDOLEN    16     U       L       SOMEFIELD  P      # Data Length example
```

Limitations:

This action requires the **U** data type. This action has no field length limitations.

### 6.1.2.2.3 Raw Data Actions

**@ Block Sequence Number Action**

This action sets the field value by adding the block count of the data, multiplied by the record's *step* value, to the record's *low* value (value = (block number * *step*) + *low*). The field value is the sequence number of the blocks within a transmission set. The block count in the transmission set is determined by the number of raw data file reads. The record's *high* value is not used when processing raw data (see Warnings below). The field value is set to the record's *low* value whenever the first block of the transmission set is being processed (the block containing first segment of data read from the raw data file).

Examples:

```
# EXAMPLE:                                   ___values___
# name length        type      action  low  high  step                 comments
SDSEQNO              15        U       @          0   0xFFF 2          # Block Sequence Number example
…
# keyword            raw data file   segment size   word fill value    comment
rawdat  tlm3204.raw  271       0x1000               # see section 6.1.3
```

In the example, the first block will have a value of zero for SDSEQNO. In each subsequent block, the value of SDSEQNO will be incremented by 2. When the transmission cycle has completed, SDSEQNO will be reset to 0.

Limitations:

This action is compatible with the **B**, **F**, **Fn**, **I**, **M** and **U** data types. This action has no field length limitations.

Warnings:

If the RID does not contain a raw data segment (rawdat keyword), this action is equivalent to the Increment (**+**) action. This is the only circumstance where the record's *high* value will be used (see section 6.1.3 - Raw Data Record Description).

## A Last Block Boolean Algebra Action

This action will set the field value to the second record value whenever the generated block is not the last block. (The question "Is this the last block?" evaluates to false.) When the last block is being generated, the field value will be set to the first record value. (The aforementioned question evaluates to true.) The last block is defined as the block generated by the last block read (end-of-file) from the raw data file (see Warnings below).

Examples:

```
#                                               values
# name length          type      action  true  false  comments
SDLASTBLK              1         U       A      1     0             # Last Block Boolean Algebra example
…
# keyword              raw data file     segment size   word fill value   comment
rawdat tlm3204.raw     271       0x1000                 # see section 6.1.3
```

In the example, when the last block of the transmission set is encountered, the field SDLASTBLK will be set to 1. For all other blocks in the set, SDLASTBLK will be set to 0.

Limitations:

This action is compatible with the **B**, **F**, **Fn**, **I**, **M** and **U** data types. This action has no field length limitations.

Warnings:

If the RID does not contain a raw data segment (rawdat keyword), this action always evaluates to TRUE, setting the field value to the first record value (see section 6.1.3 - Raw Data Record Description).

## Z Raw Data Size Calculator Action

This action will place the number of bytes read into the raw data segment of the block into the field value. The record value *offset* is a scalar value added to the number of bytes read when setting the field value.

Examples:

| # name | length | type | action | *offset* value | comments |
|---|---|---|---|---|---|
| SDSDLN | 16 | U | Z | 24 | # Raw Data Size Calculator example |

…

| # keyword | raw data file | segment size | word fill value | comment |
|---|---|---|---|---|
| rawdat | tlm3204.raw | 271 | 0x1000 | # see section 6.1.3 |

In the example, the field value will be the length (or size in bytes) of the raw data read into the 271 16-bit word raw data segment (by the rawdat keyword) plus the record value *offset*. The only block that the calculated size will be different from 566 ((271 * 2) + 24 bytes) will be the last block of the raw data file (which may not fill the entire 271 words of the segment).

Limitations:

This action requires the **U** data type and a field length of 16 bits.

Warnings:

If the RID does not contain a raw data segment (rawdat keyword), this action will set the field value to zero (see 6.1.3 - Raw Data Record Description).

### 6.1.2.2.4 Time and Date Actions

**D DOY in BCD Action**

This action retrieves the system time and puts the Day of Year (DOY) in this field formatted as a 10-bit Binary Coded Decimal (BCD) compatible with GCF and SDB header formats.

Limitations:

This action requires the **U** data type and a field length of 10 bits.

**H Time in Milliseconds Action**

This action retrieves the system time and puts the time of day in milliseconds in this field formatted as a 27-bit unsigned integer compatible with GCF and SDB header formats.

Limitations:

This action requires the **U** data type and a field length of 27 bits.

**M Time in Deciseconds Action**

This action retrieves the system time and puts the time of day in deciseconds in this field formatted as a 24-bit unsigned integer compatible with GCF and SDB header formats.

Limitations:

This action requires the **U** data type and a field length of 24 bits.

**T  Time in Centiseconds Action**

This action retrieves the system time and puts the time of day in centiseconds in this field formatted as a 24 bit unsigned integer compatible with GCF and SDB header formats.

Limitations:

This action requires the **U** data type and a field length of 24 bits.

**Y  DOY in Unsigned Action**

This action retrieves the system time and puts the day of the year in this field formatted as a unsigned integer compatible with GCF and SDB header formats.

Limitations:

This action requires the **U** data type and a field length of 16 bits.

### 6.1.3 Raw Data Record Description

Raw data records insert *segment size* static data into the stream at the same relative position that the raw data record appears in the RID file. This raw data is read in one *segment size* at a time from another file, one read for each block transmitted. The number of blocks it takes to read the entire raw data file is one transmission cycle. The raw data file can contain either ASCII or binary data.
RIDs are limited to 1 rawdat keyword per file.
General Syntax:

rawdat <raw file> <segment size> [<word fill>]

where:

**raw file** the name of the raw file to read into the blocks of the transmission set. The file has the naming convention "<name>.raw", where name has a 8 character maximum length, it must follow the normal naming conventions of the disk operating system, and all the letters must be *lower-case*.

**segment size** the size of the segment within the block which will contain the raw data. The size is in units of 16-bit words, with a valid range between 1 and 500.

**word fill** the word fill pattern. The last read operation at the end of a raw data file may not contain enough data to populate the entire segment within the block. The word fill value is used to set the remaining unset data words at the end of the raw data segment. Since the word size is 16 bits the valid range is between 0 and 0xFFFF.

Both *segment size* and *word fill* are unsigned integers and may be decimal, octal, or hexadecimal (see Table 5 - Value Representation Formats). Raw data records without a *word fill* value will use the default value of 0x1000.

Record Indices and Names:

All data and raw records use the same indexing algorithm. Record indices begin at zero and end at the record count minus 1. Each data word in a raw data segment is one record and can be referenced by index or name like a data record.
Raw records are named "RAWDAT:<m>:<nnn>", where <m> is the number of the rawdat keyword minus 1 and <nnn> is the rawdat word number minus 1 (e.g. the fifth word in the first raw data segment would be named RAWDAT:0:004).

Data records that follow a rawdat keyword are indexed by adding the segment size of the raw data to the index of the data record that preceded the rawdat keyword. The use of offsets with the bit field length (e.g. 16@4270) is unnecessary except for padding data with leading zero bits.

## 6.2 Mapping an Interface to a RID

The process for mapping a subsystem interface agreement is to a RID file is described below using a step-by-step example.

**CREATING A RID: A Step By Step Example**

TLM-3-204 (NSS-DGT) is the interface document used as the basis of the example. The interface describes the Support Data that is sent by the NSS application to the DGT application. It was chosen because it demonstrates the raw data segmenting capability as well as the bit packing/data generating capabilities of the SIV. Future versions of SIV (beyond V1.0.0) will allow a 201 protocol which will preclude the need to describe the GCF header as part of the RIDs data definition. The mapping of the GCF header is included for demonstration purposes, for those subsystems which need to process this information directly.

**STEP 1: Name the file: dgtsd.rid**

All rid files must end in the ".rid" extension.

**STEP 2: Identify the sections of data to be generated within the block**

Components of dgtsd.rid (TLM-3-204 Rapid Interface Definition)

a. GCF Header
b. Support Data Header
c. Subsystem provided NSS-DGT raw predict data
d. Checksums and GCF trailer section.

**STEP 3: Commenting the RID file**

Comments are signified with the "#". They can appear anywhere in the file, keeping in mind that any character appearing after a "#" will be considered a comment until the end of the line is reached.

**STEP 4: Define the Control and Transmission Keywords**

Define how the block will be transmitted and identify the block for the LAN software:

a. The frequency will be 10 seconds between each block transmission:

```
ssend_delay      10.0      # delay in sec. between consecutive transmissions
```

b. The default number of transmission cycles will be 1 (one set of Support Data blocks per SSEND):

```
ssend_count      1         # Number of transmission cycles per SSEND
```

c. The source process code will be the CMC:

```
proc_code_src    0xC10     # CMC Process Code
```

d. The destination process code will be the DGT:

```
proc_code_dest   0x56      # DGT Process Code
```

d. The SPCLAN to will be used to transmit:

```
select_lan       XSSPCLAN  # Selected LAN
```

e. The “no acknowledgment” protocol will be used:

```
select_protocolXSNAP   # protocol NAP
```

f. The message Id will be 0x14 because this is a support data block:

```
mid              0x14      # 890-131 ISB Message Id
```

**STEP 5: Map The Identified Sections Of The Block (Enter The Field Records Of the Block To Transmit)**

There are four sections of the block to define in this example: A GCF Header section, a Support Data Header section, a raw data section (NSS-DGT Support Data--TLM-3-204) and a checksums and GCF trailer section. These sections, except for the raw data section, will be define with Field records. The raw data section will be defined with the rawdat keyword as outlined in STEP 6.

Field records describe a field of data in the block being defined. Entry of the record is free format, separated by spaces, but a record must be contained within one line. See section 6.1.2 for a complete definition of each field of a Field Record.

This is a complete example for defining one field record. This is the Spacecraft Number field in the Support Data Header Secti on:

| | | |
|---|---|---|
| a. | provide a name for the field: | SDSCN |
| b. | specify the number of bits: | 16 |
| c. | define the data type: | U (for unsigned integer) |
| d. | define the action: | C (for constant) |
| e. | provide the values: | 77 |
| f. | provide a comment for the field: | # Spacecraft Number |

The field record should look like this:

| # name | length | type | action | values | description |
|---|---|---|---|---|---|
| SDSCN | 16 | U | C | 77 | # Spacecraft Number |

The record represents the Spacecraft Number field of the Support Data Header where 16 bits are allocated for storage of the number, the number is defined to be an unsigned integer (as described by the **U** data type) and a constant (defined by **C** action) value of 77 is the number assigned to it. Sections 6.1.2.1 and 6.1.2.2 contains a full description of all the data types and actions, respectively, supported by the SIVs RID file.

**STEP 6: Reference The Raw Data File With The "rawdat" Keyword**

After completing the field records for the GCF Header, Support Data Header and Checksum and GCF Trailer, the remaining step is identifying the data of the TLM-3-204 (NSS-DGT). The SIV allows the specification of a static file of raw data representing a Support Data interface. This is done through use of the *rawdat* keyword. The keyword syntax is:

rawdat <filename> <segment size> [<word fill>]

The raw data must appear between the last word of the Support Data Header (field 24) and the checksum word (field 296). The file which contains the raw data is *tlm3204.raw* and has a *segment size* of 271. The *segment size* is the number of 2-byte words of support data contained between the Support Data Header and the Support Data checksum.

When entering a SSEND directive, the SIV will generate a set of Support Data blocks. The number of blocks contained in the generated set depends on the number of segments it takes to read the entire raw data file. For example, if it takes 25 segments to read the entire file of raw data, then the SIV will generate 25 Support Data blocks per transmission set. If SSEND was given a parameter of C=2, the SIV will generate two sets of 25 Support Data blocks, one set right after the other--for a total of 50 Support Data blocks.

An optional word fill value can be included. The last block of a transmission set may not be entirely filled with the raw data read from the file. The fill is a numeric value used to populate the unused words of the segment (the words not containing raw data).

Examples:

| # keyword | raw data file | segment size | word fill value | comment |
|---|---|---|---|---|
| rawdat | tlm3204.raw | 271 | 0x1000 | # see section 6.1.3 |

The raw file name is "tlm3204.raw", the segment size is 271 words and it has a fill value of 0x1000. If the raw data read does not entirely fill up the segment of the last block of the transmission set (say only 100 words of raw data was read into the segment), the following 171 words will have a value of 0x1000.

**STEP 7: The Resulting RID File**

NOTE about Mapping the GCF header: If the subsystem does not need to process the GCF header or ignores it completely, then defining the individual elements of this section of the block may be unnecessary. Instead, the subsystem could opt to simply define the GCF header as a single field of data, 128 bits in length, set to zero.

For purposes of demonstration, the individual fields of the GCF header have been mapped into the RID file definition.

Figure 6-1 RID File Example

```
# TLM-3-204 Rapid Interface Definition
#
# This file describes the interface between NSS and DGT of support data as
# described in the document TLM-3-204.
#
ssend_delay      10.0                     # delay in seconds between consecutive transmissions
ssend_count      1                        # Number of transmission cycles per SSEND
proc_code_src    0xC10                    # CMC Process Code
proc_code_dest   0x56                     # DGT Process Code
select_lan       XSSPCLAN                 # Selected LAN
select_protocol  XSNAP                    # protocol NAP
mid              0x14                     # 890-131 ISB Message Id
#
# GCF HEADER INFORMATION (Words 1 through 8)
#
# name   bit length       type     action     values                 description
GCFSYNC  24               U        C          0x627627               # sync Ops67/68 (block type indicator)
GCFSRCPC 8                U        C          0xC10                  # source      (CMC)
GCFDSTPC 8                I        C          0x56                   # destination (DGT)
GCFFMT   8                I        C          0x16                   # format Ops68
GCFGDD   3                I        C          0                      # gdd
GCFUDT   7                I        C          0x41                   # udt
GCFDDT   7                I        C          0x02                   # ddt
GCFSCN   7                U        C          0x5E                   # Spacecraft Number (subsystem assigned)
GCFCSEC  24               U        T                                 # centiseconds
GCFRESV  2                I        C          0                      # reserved
GCFDOY   10               U        D                                 # day of year (DOY)
GCFBSN   12               I        +          0 0xFFF 1              # bsn
GCFDBYT  8                C        C          X                      # data byte
#
# SUPPORT DATA HEADER (words 9 through 24)
#
```

Figure 6-1  RID File Example (Continued)

```
# name   bit length       type     action     values                 description
SDSDLN   16               U        Z          24                     # Word 9: Support Data Length
SDORIG   16               U        C          0                      # Word 10: Support Data Origin
SDLASTBLK                 1        U          A                      1 0     # 11 Last Block Indicator
SDSEQNO  15               U        @          0 0x7FFF 1             # 11 Sequence Number
SDRESV2  7                U        C          0                      # 12 Reserved
SDPURGE  9                U        C          220                    # 12 Purge Date (subsystem defined)
SDCLASS  16               C        C          PR                     # 13 Class=PR/SL (subsystem defined)
SDSCN    16               U        C          77                     # 14 Spacecraft# (subsystem defined)
SDTYPE   16               C        C          G                      # 15 Type=Galileo (subsystem defined)
SDPASS   64               C        C          "      99"             # 16 pass number (subsystem defined)
SDSET    64               C        C          "NSS__DGT"             # 20 Set Name (subsystem defined)
SDREV    16               C        C          01                     # 24 Revision (subsystem defined)
#
# DGT SUBSYSTEM PROVIDED RAW DATA (words 25 through 295)
#
# The 1st ASCII line of data "**" for the NSS HEADER is contained
# in first segment of the file dgt.sdf):
#
# keyword                 raw file            size                   word fill value
rawdat   tlm3204.raw               271        0x1000
#
# name   bit length       type     action     values                 description
SDCKSUM  16               U        K          SDSDLN P               # Checksum (at word 296 for words 9-295)
```

## 6.3 RID Generation Tools

### 6.3.1 Text Editors

RID files can, and often are, created manually. The most common practice has been to copy a RID file, from another subsystem if necessary, and then edit it in any text editor. Even when using xlate (see section 6.3.4), test RID files will be needed and they will be edited versions of the base RID files created by xlate.

The RID files have the naming convention "<name>.rid", where name has a 8 character maximum length, it must follow the normal naming conventions of the disk operating system, and all the letters must be *lower-case*. Errors can be checked for using ridlint (see section 6.3.2).

The following lists some example editors:

Vi or Emacs on a Unix machine
WordPerfect (file saved using Text Only)
Word (file saved using Text Out)
MSDOS EDIT
or any text editor capable of creating a plain text file.

Files on another machine can be transferred to the SIV machine using utilities such as FTP and Kermit.

### 6.3.2 RID Lint

NAME: ridlint - SIV RID file syntax checker (a lint program for RIDs)

DESCRIPTION: This program will read all the RID files on the command line, checking each file for correct syntax.

SYNTAX: ridlint [-l] [-p -d [-v <sload-vflag>]] [<path>/]<name>.rid [...]
The command line flag arguments must precede the RID file arguments.

- -l List each file analyzed.
- -p Print the segment definition and size. The segment definition is every field record's (one per line) index, name, length, position, type and action. Position and length are given as '<bytes>:<bits>'. *
- -P Identical to -p, but with the internal values of each record printed as well. * **
- -d Print debugging output. This is the full contents of a SIV stream after the RID file is read. * **
- -v <sload-vflag> = { +as|+in, +de, +hi, +mi, +lo }.

Set the Variable actions using the SLOAD Operator Directive flags. This has no effect without specifying an output flag.

* These options produce no output when there are errors in the file.

** These options are probably only of interest to SIV developers. The outputs contain values not in the input.

All output is to standard output except the debugging output (-d) which is to standard error.

DIAGNOSTICS: Exit status is 2 for command line errors, 1 if any RID file contains errors, and 0 if no errors were found in any RID file. RID file errors include the file naming rules listed in LIMITATIONS.

LIMITATIONS Because of the limitations within SIV version 1.3.x <name> and <path> must meet the following rules:

1) <name> cannot contain uppercase letters.
2) <name> cannot be longer than 8 characters.
3) <path> cannot be longer than 66 characters.
4) Each filename must end with the ".rid" suffix.

### 6.3.3 RID Editor

NAME — ridedit.sh - Edit and error check RID files

DESCRIPTION — This will edit the RID files on the command line and then check each file for correct syntax using ridlint.

SYNTAX — ridedit.sh [ -e editor ] [<path>/]<name>.rid [ ... ]

SEE ALSO — See Section 6.3.2 - RID Lint (ridlint).

NOTES — The default editor is vi(1) or the editor named in the environment variable EDITOR. Prior to editing, the files will be saved to <pathname>.<pid>, where <pid> is the process id of the edit session. After editing, ridedit.sh will attempt to rename the backup files from <pathname>.<pid> to <pathname>.bak. If a backup file <pathname>.bak already exists, ridedit.sh will ask permission to overwrite the older copy. If the answer is no, the backup file will remain named <pathname>.<pid>. If no edits are made to a file, ridedit.sh will not leave a backup file.

This is a simple shell script aimed at beginning users. Experienced users can simply integrate ridlint into their current edit and compile methods.

LIMITATIONS — See section 6.3.2 - RID Lint - for the limitations on the RID files.

### 6.3.4 Subsystem Interface Agreement Translator

NAME

xlate - Translate Interface Definition into RID file

DESCRIPTION

XLATE converts an interface specification into a RID file. While options for XLATE control the conversion, the RID file produced is best consider a *base* file, from which many variants can be produced using the RID editor.

SYNTAX

**XLATE infile [outfile]**
**[INFORMAT=iii]**
**[DEFAULT_SELECT=aaa]**
**[INTEGER_SELECT=bbb] [FLOAT_SELECT=ccc] [CHARACTER_SELECT=ddd]**
**[PATH=ppp] [DEBUG=y/n]**

Convert the interface definition file **infile** to the *base* RID file **outfile**.

**infile** is the name of the input interface definition file. It may be a full path name. If there is no file extension, then **.id** is assumed.

**outfile** is the name of the output RID file. It may be a full path name. If not specified, then the file will be placed in the default directory and its file name will be derived from **infile** with extension **.rid**; e.g., **infile** of /path/abcd.id yields **outfile** of abcd.rid. If there is no file extension, then **.rid** is assumed.

**INFORMAT=iii** is the format of the input file. Possible values for **iii** can be:

**Standard** The new interface standard separated by tabs with fields ordered as

mnemonic
start position
type
length
name
valid values as either a range or a list
units/precision
other information

**WORD** µsoft Word table (see HTML)

**WP** WordPerfect table (see HTML)

**HTML** HTML file with fields organized as

sequence number
start position
length
data name & description
type (also called format)
units/precision
range (valid values as either a range or a list)
mnemonic

**DEFAULT_SELECT=aaa**

For fields with valid ranges, the default action code for the RID file will be **aaa**. **aaa** can be

**Constant** RID file has constant action code and first valid value (the default)

**Random** RID file has Random action code
**Sequential** RID file has Sequential action code

**INTEGER_SELECT=bbb**
For integer fields with valid ranges, the default action code for the RID file will be **bbb.**

**FLOAT_SELECT=ccc**
For floating fields with valid ranges, the default action code for the RID file will be **ccc.**

**CHARACTER_SELECT=ddd**
For character fields (i.e., ASCII, ASCII float, hex, bcd), the default action code for the RID file will be ddd.

**PATH** Interface definition files provide for REFERENCE file_name as the first two input fields where file_name points to a file which contains additional defintions for the interface. This allows interface definitions to be built up from smaller, repeatedly used definition segments. Search the default directory and then the directories listed in path **ppp** when there is an include directive. To specify multiple directories to search, separate the directory names with commas; e.g., /a/b,/usr/larry/siv/master,$JVlib.
If **PATH** is not specified, only the default directory is searched.

**DEBUG=d** Prints intermediate processing messages to standard output. This option would typically only be used by XLATE developers and testers. If **d** begins with the letter “y”, then debug printing is on, otherwise debug printing is off.

<u>EXAMPLES</u> **XLATE $ifdef/test.id**
translates the file test.id in another directory to the file test.rid in the default directory.

**XLATE test.id /usr/local/mine/siv/base DEFAULT=C FLOAT=R PATH=$library**

translates the file test.id in the default directory into base.rid in a different directory. For ranges, the default action will be C, for constant, but floating point values will have R, for random, action codes. If there are any includes in test.id, they will be searched for in the default directory and in the $library directory.

NOTES

With the exception of **infile** and **outfile**, all other arguments may be specified in any order.

**DEFAULT_SELECT** sets the default action code selection for all data types
**INTEGER_SELECT**, **FLOAT_SELECT**, and **CHARACTER_SELECT** override the
**DEFAULT_SELECT** value for the specified data types.

Environment variables and full path names may be specified for the **infile** and **outfile** and **PATH** parameters.

LIMITATIONS

The SIV user must have the appropriate read and write access to the directories and files specified by **infile**, **outfile**, and **path**.

**outfile**, if specified, must conform to the RID file name restriction of ... characters
**outfile**, if not specified, will be forced to be within the RID file name restriction of ... characters

RESPONSES

**"outfile" already exists. "Action"**
Indicates the specified output file already exists. **"Action"** is either **"Aborting"** indicating that XLATE cannot do any processing or **"File name altered to be xxxx"** indicating that the file name was adjusted to be both legal and to not have a collision with any existing file. When the file name is altered to prevent a collision, all subsequent messages containing the file name will have the altered file name.

**XLATE has successfully produced "outfile"**
Indicates successful processing

**XLATE has produced "outfile" with "nn" errors**
Indicates that errors or ambiguities were encountered in the input or that there was some problem(s) creating the output file. The output file, however, exists.

**XLATE cannot produce "outfile"**
Indicates a serious error with the input file, output file, directory permission, etc. There is no file and XLATE terminated early.

<u>REJECTIONS</u>

**Cannot find included definition file "file"**
Indicates the specified include file cannot be found in the path.

**Included file recursion error. Include ignored.**
An input file either referenced itself or a file already open for this interface definition. The offending input line is also displayed.

**"field_name" is duplicated at line "nn"**
indicates a duplicate field name.

**"field_number" "field_name" overlaps "field_number"**
indicates a start or length position value. The offending input line is also displayed.

**Illegal length value**
indicates that a specified length value is illegal or is inconsistent with the data type. The offending input line is also displayed.

**Invalid data type**
indicates an invalid data type. The offending input line is also displayed.

### 6.3.5 Subsystem Interface Agreement Definition

Pending Formal Release...

## 6.4 Changes to the RID Definition

Corrections to the manual are in plain text, while true changes to the RID definition are in underlined text.

> ***6.1 - SIV uses the Rapid*** Interface Definition (RID) to define a subsystem's interface. RID files (RIDs) are translations of Subsystem Interface Agreements (SIA) or more formally, "820-16 ; Deep Space Network System Requirements - Detailed Subsystem Interface Design". This chapter describes the RID.
>
> Section 0 defines the lexical components and syntax of the RID file. Section 6.2 provides a step-by-step guide on how to create a RID given an interface agreement. Section 6.3 describes the tools used to create a RID. Section 6.4 describes the changes to the RID definition in this release of SIV. Section 6.5 describes supplanted, but still supported, backward-compatible features.
>
> RID ASCII File Format
>
> The RID files have the naming convention "<name>.rid", where name has a 8 character maximum length, it must follow the normal naming conventions of the disk operating system, and all the letters must be *lower-case*.
>
> Value Representation Formats:
>
> Integer representations follow the standard "C" representations. A hexadecimal value has a "0x" or "0X" prefix. An octal value has a "0" prefix. A decimal value has no prefix. Commas, or any punctuation (except an optional leading "+" or "-" sign in a signed integer), are not permissible in integers (for example 10,000).
>
> Floating point representations follow the IEEE standard, using digits, a decimal point, the letter "e" or "E", and "+" or "-" sign leading the number or the letter "e". Commas are not permissible. For example, "11", "-1.1", "+.1", "1.1e2", "1e-2", and "0E+2", are all legitimate -- while "10,000.01" is not.

Text words or strings are white-space delimited sequences of characters. A text word may be a quoted string, which can contain space characters. Quoted strings use the double quote characters: "...", but the quotes are not word delimiters. The double quote character may be included in a text word by escaping the quote: \". Spaces may also be escaped, for example, 'c\ ab' is equivalent "c ab". (The single quote character is used here only for illustration and has no special meaning.)

Unless otherwise noted, keywords, field names, data types, actions, and other forms of text entry are not case-sensitive.

### 6.1.1.1 - Transmission Record Keyword Descriptions

The keywords with their limits and ranges are listed in **Error! Reference source not found.**. Ranges listed as "<integer> =" may have decimal, octal, or hexadecimal record values.

#### 6.1.1.1.1 - General Keyword

| | |
|---|---|
| ssend_count | This was *count*. |
| ssend_delay | This was *freq*. |

#### 6.1.1.1.2 - 890-131 & 890-201 Protocols Keywords

| | |
|---|---|
| block_type | The value NON was NONE. |
| proc_code_dest | This was *dst_code*. |
| proc_code_src | This was *src_code*. |
| select_lan | This was *lan_sel*. |
| select_protocol | This was *proto*. |
| sdb_data_type | This was *data_type*. |
| sub_block_id | This was *sub_id*. |

**6.1.2 - Field Record Description**

The general limitations for the field records are: 1 record per line, 255 characters per record, and 500 records per RID file.

The 500 record limit may be extended to 2999 records, by loading (SLOAD) the 500+ record RID by itself and removing (SREM) it before loading any other RIDs. SIV will enforce this restriction of only one 500+ record stream at a time.

General Syntax:

<name> <length>[@<offset>] <type> [<action>] [<value> …]

where:

| | |
|---|---|
| name | The *name* for the field record is a text word of 2 to 13 characters. The *name* must start with a letter, be unique within the RID file, and cannot be a keyword name. |
| length | The specification is "[<bytes>:][<bits>]". The maximum data *length* is 32 bytes or 256 bits. |
| offset | The specification is "[<bytes>:][<bits>]". Raw data segments and following data records are automatically positioned. Whenever an offset is used, a diagnostic or error message will be issued.<br><br>For both *length* and *offset*, the value is the *sum* of the byte and bit values. |
| type | Type limits and ranges are listed in **Error! Reference source not found.**. |
| action | Action limits and ranges are listed in **Error! Reference source not found.**. |
| value … | The number of values is dependent on the action for the field, but cannot exceed 16.<br><br>A *record value* is any value following an action in a RID file or set with the FVAL and FACT directives.<br><br>A *field value* is the value generated by SIV for that field for transmission to a subsystem. |

A data type, an action, and the action's *record values* determine the generated *field values*.

An integer record value's format (number base) determines the format in the various SIV outputs, such as the Validation report. If the values are hexadecimal, octal, or decimal, the format of the outputs will be the same.

Values that specify other field records may be field indices, field names, or the letters {P, N, E} (for previous, next, and end field, respectively.

#### 6.1.2.1 - Data Type Descriptions

The types are fully described and include bit length limits and value implementation limits. The limits are also in **Error! Reference source not found.**.

**B - Binary Coded Decimal Type**

The range of values is ( $-2^{(m \% 4)} \times 10^{(m / 4)}$, $2^{(m \% 4)} \times 10^{(m / 4)}$ ), where *m* is the field's bit length. The maximum value supported by the SIV is limited by 18 decimal digits. Record values must be decimal. Floating point BCDs are not implemented. Very large values will suffer from least significant bit loss.

**C - Character Type**

String values with lengths greater than the field length will be truncated <u>with a warning</u>. <u>String values with lengths less the field length will be silently left-justified and padded with spaces</u>. The character string will not be NUL (zero) terminated.

**F - IEEE Floating Point Type**

The field length must be 32 bits for single-precision or 64 bits for double-precision values. <u>There is no 48 bit IEEE float</u>.

**F<n> - Fixed Point Scaled Integer Type**

The range of values is [ $-2^{(m - 1 - n)}$, $2^{(m - 1 - n)} - 1/2^n$ ], where *m* is the bit length of the field. The smallest fraction is $1/2^n$. The maximum value supported by the SIV is limited by bit $m \leq 32 + n$ bits, where *n* is [ 1, 31 ]. Fractional values that are not exact multiples of $1/2^n$ will be truncated. Very large values will suffer from least significant bit loss. Negative values cannot be represented in fields larger than 32 + *n* bits.

**I - Signed Integer Type**

The range of values is [ $-2^{(m-1)}$, $2^{(m-1)}$ - 1 ], where *m* is the bit length of the field. Negative values cannot be represented in fields larger than 32 bits. Record values may be decimal, octal, or hexadecimal.

**M - Modcomp Floating Point Type**

The field length must be 32 bits for single-precision or 48 bits for double-precision values.

**U - Unsigned Integer Type**

The range of values is [ 0, $2^{m}$ - 1 ], where *m* is the bit length of the field. Record values may be decimal, octal, or hexadecimal.

### 6.1.2.2 - Action Descriptions

Limits (Limitations), potential pitfalls (Warnings), and implementation notes (Notes) are described in the Action Descriptions. Limits include the supported data types which are also in **Error! Reference source not found.**.

#### 6.1.2.2.1 - General Actions

**P ... Pick Action**

<u>This action no longer needs to be followed by a number (P<n>) indicating the number of values</u>.

**S ... Sequence Action**

<u>This action no longer needs to be followed by a number (S<n>) indicating the number of values</u>.

#### 6.1.2.2.2 - Data Block Actions

**L ... Data Length Action**

This action does not support the I (integer) or B (BCD) types .

### 6.1.3 - Raw Data Record Description

The raw data file has the naming convention "<name>.raw", where name has a 8 character maximum length, it must follow the normal naming conventions of the disk operating system, and all the letters must be *lower-case*.

All data and raw records use the same indexing algorithm. Record indices begin at zero and end at the record count minus 1. Each data word in a raw data segment is one record and can be referenced by index or name like a data record.

Raw records are named “RAWDAT:<m>:<nnn>“, where <m> is the number of the rawdat keyword minus 1 and <nnn> is the rawdat word number minus 1 (e.g. the fifth word in the first raw data segment would be named RAWDAT:0:004).

Data records that follow a rawdat keyword are indexed by adding the segment size of the raw data to the index of the data record that preceded the rawdat keyword. The use of offsets with the bit field length (e.g. 16@4270) is unnecessary except for padding data with leading zero bits.

**Errors And Warnings**

In addition to the aforementioned changes, all RID input outside the limits in the documentation, will be produce well behaved errors and error messages. Questionable input, documented in the Warnings subsection of each Action Description, will produce warning messages, but no change in behavior.

**New Tools**

The following are the RID generation tools new to this release.

### 6.3 - RID Generation Tools

#### 6.3.2 - RID Lint
ridlint - SIV RID file syntax checker (a lint program for RIDs)

#### 6.3.3 - RID Editor

#### 6.3.4 - Subsystem Interface Agreement Translator
xlate - Translate Interface Definition into RID file

#### 6.3.5 - Subsystem Interface Agreement Definition

## 6.5 Backward Compatible Features

This section describes features that will be phased out in future versions, but are still currently being supported. This includes both documented and formerly undocumented features. If a RID behavior is not listed here, or anywhere else in this chapter, it is a bug.

#### 6.1.1.1.1 - General Keyword

comm_proto This is an alias for *comm_protocol*.
count This is an alias for *ssend_count*.
freq This is an alias for *ssend_delay*.
sub_id This is an alias for *sub_block_id*.
<float> If this is the only word on the first non-comment line, it is an alias for "*ssend_delay* <float>".

#### 6.1.1.1.2 - 890-131 & 890-201 Protocols Keywords

block_type This parameter may have the following aliased values:
IFINIT = IF_INIT, NONE = NON, OPS67 = OPS_6_7, OPS68 = OPS_6_8.
data_type This is an alias for *sdb_data_type*.
dst_code This is an alias for *proc_code_dest*.

lan_sel This is an alias for *select_lan*.
proto This is an alias for *select_protocol*.
src_code This is an alias for *proc_code_src*.

**6.1.2 - Field Record Description**

General Syntax:

[<name>] <length>[@<offset>] <type> [<action>] [<value> …]

where:

name Field names are optional.

**6.1.2.2.1 - General Actions**

**+ ... Increment Action**

This may have only one record value. This one value will be the *low* value, with the *high* and the *step* values set to 32767 (0x7FFF) and 1, respectively.

***- ... Decrement Action***

This may have only one record value. This one value will be the *high* value, with the *low* and the *step* values set to 0 and 1, respectively.

***P ... Pick Action***

If this action is followed by a number (P<n>), the number <n> will be used to test the number of arguments and produce a warning if there are more or less than <n>. (Originally <n> was required to determine the number of arguments.)

***S ... Sequence Action***

If this action is followed by a number (S<n>), the number <n> will be used to test the number of arguments and produce a warning if there are more or less than <n>. (Originally <n> was required to determine the number of arguments.)

**6.1.2.2.3 - Raw Data Actions**

**@ ... Block Sequence Number Action**

This may have only one record value.  This one value will be the *low* value, with the *high* and the *step* values set to 32767 (0x7FFF) and 1, respectively.

# 7

# USER'S GUIDE

This section provides the software developer or tester with procedural information on using the SIV within a test laboratory.

## 7.1 Configuration

The SIV can be configured to run within your personal test lab or from the JPL DTF-21 lab.

### 7.1.1 SIV In Your Development Lab

Currently, SIV software supports Sun Sparc 10 running Solaris 2.x - Open Windows is not required, the X11 interface will run in any X11 environment.

Software can be acquired from either JPL SPMC, ISDS CM, or from SIV development personnel. You may elect to obtain tested software from JPL SPMC or ISDS CM or the latest 'available' release from SIV development personnel. As of the end of FY95, Build 3, Version 1.3.1 was the latest software delivered to JPL.

#### 7.1.1.1 SIV Software Installation

This information also comes in a file named ReadmeSiv.txt that comes with the SIV distribution. The readme file will always contain the most up-to-date information.

@(#) ReadmeSiv.txt 1.6 - 12 Sep 1995 07:00:01 (ISDS DSN-DSDPC JPL)

Installation and Usage Notes for the Subsystem Interface Verifier (SIV)

John Veregge - veregge@isds-server.jpl.nasa.gov - (818) 584-0878 x109
Information Systems Development Support (ISDS) Team
(DSN Data Systems Development Program Contract to JPL)

This file contains user documentation in the USAGE NOTES section and subsection 6 in the INSTALLATION REQUIREMENTS section. The latter covers the installation of new user working directories using the supplied makefile. The remaining sections are installation related and intended for the system administrator installing SIV.

These notes are not a substitute for the SIV User's Guide and Software Operator's Manual (SOM). Contact SPMC for a copy of UG-DOI-5249-TP-B.

Figure 7-1 ReadmeSiv.txt

## INSTALLATION REQUIREMENTS

1. Solaris 2.x running on a Sun. Sparc 10+ is recommended, not required.

2. The IPC module limits must be set in the file /etc/system.

3. The LAN file /dev/le must be set up correctly.

4. The working directories must reside on a local disk.

5. The public domain program sudo(8) must be installed.

6. The SIV system directory should be installed in the usual root-owned third-party software location. Then the supplied makefile should be used to create user-owned working directories.

*The following notes are an elaboration on requirements 2 through 6.*

1. Although the SPARC 10 is not required, it is the current test platform. Smaller machines (LX, classic, etc.) may require smaller settings in /etc/system.

2. The root file /etc/system must have the following settings:

```
set semsys:seminfo_semmap = 40
set semsys:seminfo_semmni = 200
set semsys:seminfo_semmns = 400
set semsys:seminfo_semmsl = 100
set shmsys:shminfo_shmmax = 4000000
set shmsys:shminfo_shmmni = 600
set shmsys:shminfo_shmseg = 32
```

   These entries change the kernel and require rebooting the machine.

3. The root file /dev/le must have the following permissions:

```
permissions owner group filename
----------- ----- ----- ---------------------------------------
lrwxrwxrwx  root  root  /dev/le -> ../devices/pseudo/clone@0:le
crw-------  root  sys   /devices/pseudo/clone@0:le
```

4. SIV and Multiuse Software (MSW) cannot run on a network mounted directory (Network File System (NFS), Andrew File System (AFS), etc.). When running SIV, the working directory must reside on a local disk. Slower disk access from NFS causes MSW to time out and lose task synchronization during initialization (SIV is built on top of MSW.)

5. SIV requires root access via the public domain program sudo(8). SIV will fail if sudo cannot be found or the user is not on the sudo list.

For the older sudo(8) versions, the user entries in /var/adm/sudoers must be at least:

<user name> msw.sh cleanup_msw.sh

or

<user name> ipcrm kill msw.sh cleanup_msw.sh

For the more sophisticated Univ. of Colorado sudo(8) version 1.3 or greater, the user entries, entered with visudo(8), must be at least:

Host_Alias  SIVHOSTS=hostname1,hostname2,hostname3

with

Cmnd_Alias  SIVCMDS=/sivroot/path/bindir/
<user name> SIVHOSTS=SIVCMDS

or

Cmnd_Alias  SIVCMDS=/bin/ipcrm,/bin/kill,/sivroot-path/bindir/
<user name> SIVHOSTS=SIVCMDS

The University of Colorado sudo(8) is available by anonymous ftp from

ftp.cs.colorado.edu
/pub/sysadmin/utilities/cu-sudo.x.y.z.tar.Z

6. Set up the SIV root directory in any standard place, owned by root or the user id you use for third party software installations.

   Use the makefile Sivuser.mak to create working directories for users. Do not execute this makefile as root. Have each user run the makefile to create their own working directories. The user directory will contain symbolic links to the required files in the SIV home directory.

   ```
   % cd ~
   % mkdir siv
   % cd siv
   % make -f /opt/siv/Sivuser.mak SIVROOT=/opt/siv
   ```

   Optionally you can set the two ethernet addresses SIV requires by adding the following macro definitions on the make command line.

   ```
   % make -f -f /opt/siv/Sivuser.mak SIVROOT=/opt/siv SIVHOST=<hostname> TGTHOST=<hostname>
   ```

   SIVHOST is the computer SIV will be run on from this working directory (defaults to the output of ‘uname -n‘) and TGTHOST is the computer running the subsystem SIV will be testing (no default value).

   This makefile also creates a symbolic link to the script siv.sh. Siv.sh can be copied and edited as necessary - see EDITING SIV FILES. However, unless the path to sudo(8) needs to be added to $PATH, siv.sh shouldn’t need editing.

# USAGE NOTES

## START-UP

SIV is built on top of Multiuse Software. You don't need to know Multiuse to use SIV. However, you have access to all the Multiuse functionality (Operator Directives, Displays, etc.) within SIV.

The file to start SIV is the script file siv.sh. SIV is fully up and running after it reports the local terminal is enabled. To terminate SIV, type in "term abort". After SIV terminates, the script siv.sh, will completely cleanup after itself. The script siv.sh completely replaces the script files go.sh and cleanup.sh.

## SIV TERMINAL (USER INPUT)

The SIV terminal has the prompt '>', but occasionally gets confused by output messages and you may see no prompt or many (i.e. ">>>>"). Don't worry, as it's only the prompt that bunches up. SIV will have no trouble reading what you type in regardless of the state of the prompt.

The SIV terminal is primarily for input, however, status messages relating to your input and help messages will be sent there as well.

Help for all commands or Operator Directives (OD) is in the form:

| | |
|---|---|
| > <command> ? | - Tell me what this command does. |
| > <command> ?? | - Show me this command's usage and parameters. |

**SIV SCREENS (SIV OUTPUT DEVICES)**

Output is handled by SIV screens, that you must initialize. A SIV screen uses a terminal and understands the following terminal devices:

<terminal-type>: ANSI, PSITECH, TEK, VT100, WY370, XPSI *.

* XPSI is a special case. It is a X11 Window, not a terminal. The device-file parameter is the X11 display id (i.e. hostname:0.0).

All the screens are initialized the same way:

> TERM SCREEN <number> ABORT
> TERM SCREEN <number> <terminal-type> <terminal-device-file>

There can be 1 to 4 (<number>) SIV screens. Each SIV screen takes over the specified terminal device, turning it into an output only window. The preceding "abort" line is necessary to clear out any previous terminal-type definition for that screen. The abort line can also be used to return the terminal to it’s normal non-SIV use.

For example, to create a SIV screen on your PC, you need at least 2 VT100 remote terminals running remotely from the Sun running SIV. One VT100 has the SIV terminal running on it (you are running siv.sh). The other can be turned into a SIV screen by getting the terminal device of the VT100 window using tty(1). Take the device pathname returned by typing "tty" into the free terminal, for example "/dev/pts/1", and use it as the terminal-device-file argument:

> TERM SCREEN 1 ABORT
> TERM SCREEN 1 VT100 /dev/pts/4

This will initialize your free VT100 terminal as the SIV output device. However, this is only the device initialization. To see information, you must tell SIV which of it’s displays to use screen number 1.

**SIV DISPLAYS (SIV OUTPUT)**

To start a SIV display on an initialized SIV screen device, you must tell SIV which of it’s displays to use the screen.

> D <display-name> <screen-number><screen-quadrant>

As an example, to see the SIV status display on screen 1:

> D STS 11

The final parameter of "11" is actually two numbers. The first digit is the screen-number. The second digit is the screen-quadrant. Some displays are 1/2 screen width and can go in all four screen-quadrants. Full width displays can only go in quadrants 1 (upper) and 2 (lower).

ALL SIV displays should use quadrant 1, when using vt100. When using any other screen device, two full-width displays will fit each screen.

Some displays are larger than one page and are scrollable. Scrolling is controlled by:

> VIEW <display-name> {F|B} [<page-count>] # forward | back
> VIEW <display-name> {U|D} [<line-count>] # up | down
> VIEW <display-name> {L|S} <line|for> # goto line | search for

The SIV software has five software displays and nine help displays. (In addition to the standard Multiuse Software displays.)

| Display | Width | Scroll | Description |
|---|---|---|---|
| CNF | full | yes | SIV configuration and RID file list |
| ISB | full | yes | od(1) like view of a data block/segment |
| RID | full | yes | the field values of the current stream |
| STRM | full | yes | the list of the loaded or active streams |
| STS | half | no | status of the SIV software |
| FNCAP | full | yes | general introductory help |
| HDIR | full | yes | (Help) operator DIRectives reference |
| HDIS | full | yes | (Help) DISplays reference |
| HEVT | full | yes | (Help) EVenT notices or error messages |
| QDIR | full | no | (Quick) DIRectives ref. (one-page list) |
| QDIS | full | no | (Quick) DISplays ref. (one-page list) |
| UPDAT | full | yes | New SIV features or UPDATes. |

* Software Operator's Manual (SOM) See the SIV User's Guide for more information.

**EDITING SIV FILES**

If you need to edit SIV files, all you need to is rename the symbolic link and then copy the SIV file/s to the SIV working directory.

```
% cd siv
% mv cnfdir cnfdir.orig
% cp -pr cnfdir.orig cnfdir
% mv sysinit.ini sysinit.ini.orig
% cp -p sysinit.ini.orig sysinit.ini
```

DO NOT DO THIS AS ROOT! The SIV working directory has been separated from the SIV root or system directory to limit the damage caused by the inevitable accidents. Make all your changes from the user-owned working directory. The working directory is set up with the makefile, Sivuser.mak, in the SIV system directory.

All the symbolic links can be recreated by deleting or renaming a file and running make again.

**ARP & ETHERNET ADDRESSES**

If you are using arp(1) to get ethernet addresses for SIV configuration use the supplied script, arp.sh, instead. The script arp.sh uses arp(1), but will reformat the number into the format SIV requires.

```
% arp mufasa
mufasa (137.79.115.85) at 8:0:20:23:1:86 permanent published
% bindir/arp.sh mufasa
080020230186
```

**CONFIGURING SIV**

A configuration file (CNF) must exist for your subsystem which contains LAN connectivity information required for establishing communication between the SIV and your target machine. The CNF naming convention is "<ddc>.cnf", where <ddc> is your 3 or 4 character mnemonic used for Monitor and Control. CNF files are located in the SIV subdirectory ./cnfdir. To create a new file, copy an existing file and edit it.

The two ethernet addresses in the CNF are important and easy to get wrong. Use the supplied script arp.sh (see above) to obtain these values. The hexadecimal numbers in the CNF cannot have leading "0x".

**GENERATING DATA**

| | |
|---|---|
| CNF | Configure the SIV |
| SLOAD | Load a Stream into Control Table |
| SSEND | Send a Stream to the Target |
| SREM | Remove a Stream from Control Table |

There is an upper limit of 6 streams loaded with SLOAD. A stream must be loaded into the control table before it can be sent to the target.

**VALIDATING/VIEWING DATA**

| | |
|---|---|
| DUMP | Dump ISB Message Blocks |
| LOG | Log Inbound Streams to Disk |
| VAL | Control Validation for a Given Stream |

You must enable logging on a stream before you validate that stream because the validation reads the log file.

The validation on a stream disables itself on reaching the end of the log file. Unless you wish to type in "val <stream> e" a great deal, wait until after all the blocks your are expecting have arrived.

Validation is the only way to view interpreted data. DUMP saves images of the blocks. Only VAL uses the RID to print the data in a human readable format.

### 7.1.1.2 Configuring Data Files used by SIV

A configuration file (CNF) must exist for your subsystem which contains LAN connectivity information required for establishing communication between the SIV and your target machine. The naming convention for CNF files is '<ddc>.cnf', where <ddc> is your 3 or 4 character identifier used for Monitor & Control. CNF files are located in directory ./cnfdir/. Edit an existing or create your own CNF file using the following steps:

STEP 1: Verify/enter your subsystem DDC (3 to 4 characters).

STEP 2: Verify/enter your logical process code in hexadecimal, but without a leading "0x".

STEP 3: Verify/enter the target machine and SIV machine ethernet addresses for your test lab LAN. The included shell script ./bindir/arp.sh will give you the address in the correct format. The correct format is 12 hexadecimal digits, including leading zeros, but without a leading "0x".

STEP 4: Verify/enter the pathname to your Rapid Interface Definition (RID) files. If files were provided with SIV, the pathname should be ./riddir/<ddc>. Alternately, a full path can be specified here to identify where on the SIV machine you have placed your RIDs. Section 6 describes the format for RIDs.

```
#********************************************************************************
# Information necessary for FAT table transfer; LAN addresses are 48 bit hexadecimals.

TGT_DDC         MPA ;              # 1 to 4 character mnemonic
TGT_PRCCODE     9b0 ;                    # process code of subsystem - hex range [0, fff], without a leading 0x
TGT_SPCADDR     00004b0a2dc0 ;           # ethernet address of subsystem - 12 hex digits, incl. leading zeros
SIV_SPCADDR     00004b099cb8 ;           # ethernet address of SIV - ditto
RID_LOC         riddir/mpa ;       # may be relative or absolute pathname
```

Figure 7-2  Configuration File Example

#### 7.1.1.3 Activating SIV Software

SIV can be activated by entering "siv.sh" from within your siv working directory. The following messages will result:

```
 sysinit main() <doy/time>:   0 CSW Version MSW ICDS CM Build x.x.x <date>
 cstcts: err exit <cannot initialize; not getting serial data from TCT; errno 0
 Mon_n_Ctl_Task  Date: <date>

 sysinit: sysinit: task cstcts failed to ack level 256 within 30 sec
 main() stf <doy/time>:   311 MSW Version: MSW ISDS CM Buildx.x.x <date>
 main() stf <doy/time>:   320 STF Args: IRT:999, LDP:99, Spclan, nohrlan, P:0
 spinit() stf <doy/time>:  320 Lastime.run data is too old:  <nnn> Min ago.
 CSmainlp()  Mon_n_Ctl_Task <doy/time>:   5555 Timeout is 200 milliseconds
 SIV - Subsystem Interface Verifier
 DOI-5249-TP-A Vx.x.x <date>
 sysinit:  sysinit: cstcts task failed to ack level 256
 sysinit: Task activation completed (25 tasks out of 26)
 progid rawgen <doy/time>:    0 DOI-5249-TP-A  Vx.x.x <date>
 progid rawgen <doy/time>:    0 BUILD DATE <day> <mon> <time> PDT <year>
 progid datagen <doy/time>:    0 DOI-5249-TP-A  Vx.x.x <date>
 progid datagen <doy/time>:    0 BUILD DATE <day> <mon> <time> PDT <year>
 progid isb_dmp <doy/time>:    0 DOI-5249-TP-A  Vx.x.x <date>
 progid isb_dmp <doy/time>:    0 BUILD DATE <day> <mon> <time> PDT <year>
 progid sivmgr <doy/time>:    0 DOI-5249-TP-A  Vx.x.x <date>
 progid sivmgr <doy/time>:    0 BUILD DATE <day> <mon> <time> PDT <year>
 Local terminal is ENABLED.
<doy/time> W!  700:OFFLINE EN MSGS OCCURRED!! SEE DISPLAY 'ENOFF'
```

Figure 7-3  SIV Session Starup Messages

#### 7.1.1.4 Terminating SIV Software

SIV is terminated by typing "term abort". The SIV can also be terminated by the 'kill' key defined for your login. Typically this is 'CTRL-C' or 'CTRL-T'.

#### 7.1.1.5 Running Multiple SIV Machines

This can be accomplished as each machine hosting SIV application software will have it's own ethernet address. In addition, FAT LOCKing is performed by the SIV which effectively eliminates the potential process code conflicts over the same LAN.

Section 7.2 addresses the different SIV operating modes relates to this topic.

### 7.1.2 SIV at the JPL Development Test Facility - DTF-21

Currently SIV is on a Sun Sparc 10 Solaris 2.3 machine machine at DTF-21.
**Equipment Scheduling**

SIV equipment scheduling follows the normal DTF-21 scheduling process. The form is accessible at the DTF-21 office or can be electronically submitted via Email (DTF-21 is on CCMAIL). This request must be submitted before noon on Wednesdays when scheduling time for the following week.

### 7.1.2.1 What You Need at DTF-21

The following is a check list to help you prepare for DTF-21 testing:

[ ] Schedule equipment as described above
[ ] Bring you application test software (if applicable)
[ ] Bring your updated RIDs on 8mm, DC6150 Data Cartridge Tape (or compatible)
[ ] Bring pen & notebook for taking notes
[ ] Blank 8mm, DC6150 Data Cartridge Tape for bringing back test logs

Once you arrive at DTF-21 you will need to do the following.

[ ] Check in with the DTF-21 lead for that shift and borrow the SIV documentation.
If you have not been there before, get the password to one of the group accounts on the SIV machine.
[ ] Log into the SIV machine.

If the SIV working directory has not been set-up or you want a new directory, perform the following steps.

[ ] Make a working directory with mkdir(3) and cd(3) into it.
[ ] Run make(3) using the makefile described in Item 6 on Page 4, Sivuser.mak.
make -f /data/sivroot/sivroot-1.3.x/Sivuser.mak SIVROOT= data/sivroot/sivroot-1.3.x
where x is the revision index, usually bug fix revsions. As of end of FY96 the SIV version was 1.3.1.

[ ] Start up SIV using siv.sh and begin...

## 7.2 SIV Operating Configurations

The SIV can be operated in one of two configurations--traditional and link-oriented.

### 7.2.1 Traditional Configuration

In this configuration, SIV operates as the CMC, LMC, and all other assemblies with whom the target subsystem has interfaces. It involves only two machines--SIV and the target (TGT) assembly. In this role, SIV is responsible for generating the FAT blocks and sending the required Configuration Change Notifications (CCNs) to the TGT. It is the traditional configuration initially envisioned for SIV and is anticipated to be the most often used in support of interface development and test.

FAT generation is performed when the SIV "CNF" directive is entered. The FAT transmission protocol establishes the SIV ethernet address to be assigned to the process codes of all interfacing subsystems to the TGT--including CMC and LMC by default. The FAT LOCK capability is utilized to protect against communications potentially initiated by other assemblies sharing the same LAN. Logical Data Paths are established as part of this configuration process. Once locked, only the SIV can reconfigure to release the TGT without restarting that assembly's application software.

The SIV can also transmit CCNs to the TGT. This is accomplished by using the SIV's data generation capability on a RID which defines the CCN for this particular TGT. The typical SIV directive entry scenario for CCN transmission is:

```
SLOAD tgtccn
STRM sac=1                    .. quiescent -> unassigned mode
SSEND
(await TGT processing completion)
STRM sac=2                    .. unassigned -> assigned mode
SSEND
```

Control over when CCNs are generated is thus given to the tester. This allows for intermediate actions to be performed at the TGT as may be needed in test situations.
Once communication paths have been established, the SIV is now ready to accept and forward directives destined to the TGT.

This is accomplished using the 'TGT' SIV directive. As such, **ALL** test entries can be performed from the SIV local terminal.

Two forms of test are now envisioned:

SIV generation of data streams to the TGT (emulation of all other interfacing assemblies)

SIV reception of data streams sent from the TGT (destined to any of its interfacing assemblies)

SIV directives enable the tester to further filter data flows to test simple 'single stream' tests or to run 'multi-stream' tests.

### 7.2.2 Link-Oriented Configuration

In this configuration, the tester wishes to use SIV to emulate a single assembly which is unavailable while maintaining the DMC as the Monitor and Control subsystem. As such, the SIV is included in the link and does **NOT** perform any DMC emulation functions. To configure SIV for this mode, the following must be performed:

STEP 1: At the CMC, the assembly for which SIV will emulate must have its ethernet address changed to that machine on which SIV software is hosted.

STEP 2: The SIV "NOFAT" parameter is entered with the SIV "CNF" directive. Note that the RID files for a SIV configured to emulate all the subsystems for the target are different for a SIV configured to emulate the target.

STEP 3: The CMC directives to configure a link are entered including the DDC for the emulated assembly (i.e., do NOT use "SIV").

STEP 4: The SIV "SCOM" directive may be used to put the SIV into assigned mode. This enables SIV directive entry from the LMC.

In this mode, the SIV will NOT send out FATs. It can be used to send or receive data streams for interfaces between the emulated assembly and any other assemblies.

**NOTE: This test can be performed at the DTF-21. A word of caution: allow DTF-21 personnel to make all ethernet address changes and remind them to return the proper values when your test time is over.**

## 7.3 Automated Interface Testing with Auto-Tester

SIV provides the MSW script language service *Auto-Tester* for providing automated testing capabilities. The Operator directive that controls the scripts is ACTL and is documented in the MSW SOM, Section 2. The Auto-Tester script language definition is documented in the MSW UG, Volume 2, section 1.

The inputs to the Auto-Tester are called "Test Scripts". A test report is always generated and is referred to as the "Test Log". This section describes these files and provides some direction for their usage for interface testing.

### 7.3.1 Examples of Auto-Tester Use

Using the MSW Auto-Tester service integrated into the SIV enables test repeatability as well as test record keeping. To order to provide an idea of the power of the Auto-Tester, the following lists certain test capabilities of interest to interface testing:

<u>Testing Data Blocks Inbound to Your Subsystem (i.e., outbound from SIV)</u>:

- generation of full range of data values, both valid and erroneous
- vary individual values while maintaining others constant for concentrating on specific data items
- generating data value combinations which instigate specific actions by your subsystem
- vary data values conditionally based on an individual or combination of data values sent out by your subsystem

Testing Data Blocks Outbound from Your Subsystem (i.e., inbound to SIV):

- control which data stream is being validated without affecting your subsystems data generation process
- control validation filters based on data contents
- clear reception parameters to re-baseline block counts or validation reports generated
- control raw data logging based on data contents

Also note that tests can be conducted which interrelate inbound and outbound SIV streams (e.g., an inbound block received may trigger the generation of a outbound block).

### 7.3.2 Test Script Format

MSW documentation describes the format and commands available for test scripts. The following lists some simple commands that can aid in immediate use of the Auto-Tester.

| | |
|---|---|
| MSG <text> | generates an event message which is also written to the test log |
| OD <directive> | submit <directive> to be executed / directive is expected to complete successfully |
| ODREJ <directive> | submit <directive> which is expected to fail |
| WAIT <seconds> | specifies an amount of time to delay before executing the next test script commands (.1 - 999999) |
| SUSP [<text>] | generates optional message and waits indefinitely for tester to enter a Auto-Tester resume command |
| CALL <file> [<parms>..] | execute another test script and terminate |
| CHAIN <file> [<parms>..] | execute another test script and return to complete the current test script |

The Auto-Tester test script language also supports logic control constructs, such as:

| | |
|---|---|
| SET <variable> <value> | enables update of shared memory values & supports value comparisons, etc. |
| IF-ELIF-ELSE-ENDIF | provides conditional logic to command execution; provides for shared memory/data block interrogation |
| LOOP-ENDLOOP | provides for repeated execution of a group of test script commands |

```
#*******************************************************
#    File:       rid.tst
#
#    Description:Script for testing the SIV
#                capability to generate data streams based
#                upon 'rid' files.  This also demonstrates
#                the capability to dynamically modify the
#                'rid' definition via operator directives.
#
#    Usage:      Test Case x
#
#*******************************************************

DELAY    .1
ODTST    E
EVT      ALL      E
OD       DBG      od  e

MSG      On TGT> Activate
MSG      displays FULL1 and HALF1
SUSP

DREP     HALF1    STS
DREP     FULL1    RID

# Default values
OD       SH cat riddir/test/constant.rid
OD       SLOAD    constant
MSG      On TGT> Verify RID
MSG      display matches 'constant.rid'
SUSP

# Verify initial values remain constant
OD       SSEND    c=6 f=5
WAIT     30

# Modify current values
OD       FVAL     string  -1
OD       FVAL     integer  -1
OD       FVAL     unsigned  1
OD       FVAL     mfloat  -1
OD       FVAL     fixedpt  1
OD       FVAL     bcd  1
```

```
MSG      On TGT> Verify RID
MSG      display values = +/- 1
SUSP

# Verify new values remain constant
OD       SSEND    c=3 f=10
WAIT     30
# Modify actions
OD       FACT     unsigned V       32768 65735 1
OD       FACT     integer  R       -256 256
OD       FACT     string   P4      str1 str2 str3 str4
OD       FACT     char     S6      a b c d e f
OD       FACT     sfloat   -       -2.2  2.2  .5
OD       FACT     dfloat   +       -123.123 321.321 100
OD       FACT     bcd      +       123 800 123
# Time Action Fields
OD       FACT     centisec C       100      #  24 bits
OD       FACT     decisec  C       10       #  24 bits
OD       FACT     bindoy   C       321      #  16 bits
OD       FACT     bcddoy   C       123      #  10 bits
# Checksum Action Fields
OD       FACT     fieldck  C       0x0a0a   #  16 bits
OD       FACT     blockck  C       0xFaFa   #  16 bits
# Byte Length Action Field
OD       FACT     length   C       32767
MSG      On TGT> Verify actions
MSG               on RID display
SUSP

# Verify new actions / post-test analysis
OD       SSEND    c=15 f=5
WAIT     30

MSG      "Anomalous Conditions"

# Missing .rid file
ODREJ    SLOAD    none

# multiple errors in '.rid' file
ODREJ    SLOAD    errors

# cleanup
OD       SREM     constant

MSG      Print file 'constant.val'
MSG      for post-test analysis
MSG      "RID Test Complete"
```

Figure 7-4  Auto-Tester Script Example

### 7.3.3 Test Logs

Use of the output logs generated by the Auto-Tester should be considered during generation of a test script. The test logs generated contains test script commands, messages, event messages, directive responses, etc. Test scripts can be created in a way that their execution can be performed overnight with results taken from the corresponding test log file generated. This is also beneficial for regression-type testing.

MSW documentation describes the specific information and format for test logs.

154 22:30:08.0: ODRX: ACTL RID
154 22:30:08.0: CMD: DELAY .1
154 22:30:08.0: CMD: ODTST E
154 22:30:08.0: CMD: EVT ALL E
154 22:30:08.0: OD-C: DBG OD E
154 22:30:09.0: RS-C: DBG MODE 'OD': ENABLED
154 22:30:09.0: MSG: On TGT> Activate
154 22:30:10.0: MSG: displays FULL1 and HALF1
154 22:30:10.0: SUS: ACTL SUSPENDED, Issue 'ACTL RESM'
154 22:30:50.0: ODRX: ACTL RESM
154 22:30:50.0: RSP: ACTL resumed
154 22:30:50.0: CMD: DREP HALF1 STS
154 22:30:50.0: CMD: DREP FULL1RID
154 22:30:50.0: OD-C: SH CAT RIDDIR/TEST/CONSTANT.RID
154 22:30:52.0: RS-C: SHELL COMMAND STATUS: 0
154 22:30:52.0: OD-C: SLOAD CONSTANT
154 22:30:54.0: RS-C: SLOAD OF CONSTANT COMPLETE
154 22:49:23.0: OD-C: SSEND C=3 F=1
154 22:49:23.0: EVT: PA 040:STREAM constant STARTED
154 22:49:23.0: RS-C: SSEND.
154 22:49:24.0: OD-C: FVAL STRING -1
154 22:49:25.0: RS-C: STRING=-1
154 22:49:26.0: OD-C: FVAL INTEGER -1
154 22:49:26.0: EVT: PA 043:STREAM constant ENDED
154 22:49:26.0: RS-C: INTEGER=-1
154 22:49:27.0: OD-C: FVAL UNSIGNED 1
154 22:49:27.0: RS-C: UNSIGNED=1
154 22:49:28.0: OD-C: FVAL SFLOAT -1
154 22:49:28.0: RS-C: SFLOAT=-1.000000
154 22:49:29.0: OD-C: FVAL DFLOAT -1
154 22:49:29.0: RS-C: DFLOAT=-1.000000
154 22:49:31.0: OD-C: FVAL CHAR 1
154 22:49:32.0: RS-C: CHAR=1
154 22:49:32.0: OD-C: FVAL BCD 1
154 22:49:33.0: RS-C: BCD=1.000000
154 22:49:33.0: MSG: On TGT> Verify RID
154 22:49:34.0: MSG: display values = +/- 1
154 22:49:34.0: SUS: ACTL SUSPENDED, Issue 'ACTL RESM'
154 22:54:39.0: EVT: PA 040:STREAM constant STARTED
154 22:54:50.0: EVT: PA 043:STREAM constant ENDED
154 22:55:17.0: ODRX: ACTL RESM
154 22:55:17.0: RSP: ACTL resumed
154 22:55:17.0: OD-C: SSEND C=3 F=10
154 22:55:18.0: EVT: PA 040:STREAM constant STARTED
154 22:55:18.0: RS-C: SSEND.
154 22:55:19.0: OD-C: FACT UNSIGNED V 32768 65735 1
154 22:55:20.0: RS-C: UNSIGNED: ACT=V VALS=[32768, 65735, 1]
154 22:55:20.0: OD-C: FACT INTEGER R -256 256
154 22:55:21.0: RS-C: INTEGER: ACT=R VALS=[-256, 256]

Figure 7-5 Auto-Tester Log Example

154 22:55:21.0: OD-C: FACT STRING P4 STR1 STR2 STR3 STR4
154 22:55:22.0: RS-C: STRING: ACT=P VALS=[STR1, STR2, STR3, STR4]
154 22:55:22.0: OD-C: FACT CHAR S6 A B C D E F
154 22:55:23.0: RS-C: CHAR: ACT=S VALS=[A, B, C, D, E, F]
154 22:55:34.0: OD-C: FACT FIELDCK C 0X0A0A
154 22:55:34.0: RS-C: FIELDCK: ACT=C VALS=[2570]
154 22:55:36.0: OD-C: FACT BLOCKCK C 0XFAFA
154 22:55:37.0: EVT: PA 043:STREAM constant ENDED
154 22:55:37.0: RS-C: BLOCKCK: ACT=C VALS=[64250]

154 22:55:38.0: OD-C: FACT LENGTH C 32767
154 22:55:38.0: RS-C: LENGTH: ACT=C VALS=[32767]
154 23:07:37.0: OD-C: SSEND C=15 F=2
154 23:07:37.0: EVT: PA 040:STREAM constant STARTED
154 23:07:37.0: RS-C: SSEND.
154 23:07:38.0: MSG: Anomalous Conditions
154 23:07:38.0: OD-R: SLOAD NONE
154 23:07:39.0: EVT: LO 031:FAILED TO OPEN FILE riddir/test/none.rid.
154 23:07:39.0: RS-R: FAILED TO OPEN FILE RIDDIR/TEST/NONE.RID.
154 23:07:39.0: OD-R: SLOAD ERRORS
154 23:07:40.0: EVT: LO 031:FAILED TO OPEN FILE riddir/test/errors.rid.
154 23:07:40.0: RS-R: FAILED TO OPEN FILE RIDDIR/TEST/ERRORS.RID.

154 23:07:41.0: MSG: Print file 'constant.val'
154 23:07:42.0: MSG: for post-test analysis
154 23:07:42.0: MSG: RID Test Complete
154 23:07:43.0: STS: ACTL SEQUENCE COMPLETED: rid.tst :
154 23:07:43.0: STS: ACTL rid.log COMPLETED

Figure 7-5 Auto-Tester Log Example (Continued)

**THIS PAGE INTENTIONALLY LEFT BLANK**

# APPENDIX A

# NEW FEATURES

NEW OR CHANGED CAPABILITIES TO SIV VERSION 1.2.5

## New Operator Directives

| | |
|---|---|
| D201 | initialize the 890-201 stream routing tables. |
| LOG | log inbound streams. |
| RAWP | change the raw data file directory path. |
| RIDP | change the RID File directory path. |
| SCOM | change the communication mode for SIV. |
| TGT | send operator directives to the target. |
| VAL | validate logged inbound streams. |
| XPSI | generate X11 screens for the subsystem. |

## Changed Operator Directives

| | |
|---|---|
| CNF | added NOFAT and LINK parameters. |
| DFILE | added raw image block dumps - see DUMP. |
| IDUMP/ODUMP | replaced by DUMP. |
| SHDIR | replaced by STRM. |

New RID Keywords

| | |
|---|---|
| comm_protocol | identifies when 201 encapsulation is required. |
| data_begin | marks the beginning of the data portion of 890-201 data. |
| data_end | marks the end of the data portion of 890-201 data. |
| data_type | identifies the 890-201 Standard Data Block (SDB) data type. |
| sub_id | 890-132 Monitor Segment Number or<br>890-201 data type 41 NOCC realtime packet identifier. |

## NEW OR CHANGED CAPABILITIES TO SIV VERSION 1.3.1

New Online Help (using standard MSW displays)

| | |
|---|---|
| fncap | help page on beginning SIV usage. |
| hdir | help page on operator directives. |
| hdis | help page on displays. |
| hevt | help page messages and event notices. |
| updat | help page on new (updated) features. |

New Tools

| | |
|---|---|
| siv.sh | SIV startup script. |
| msw.sh | multiuse startup script, replaces go.sh and cleanup.sh, called by siv.sh. |
| cleanup_msw.sh | support script called by msw.sh. |
| ftok | support program called by cleanup_msw.sh. |
| ridlint | RID file syntax-checker. |
| xlate | 890-16 interface definition to RID file translator. |
| ridedit.sh | RID file editor/syntax-checker for beginning users. |
| arp.sh | utility that returns the ethernet addresses in the format required by SIV. |
| Sivuser.mak | makefile for creating user working directories |
| ReadmeSiv.txt | readme file covering installation and beginning usage. |

The SIV operational directory was redesigned, so that all the system files are installed in a standard third party software location and multiple user working directories are installed away from the system files with minimal diskspace usage and the level of file sharing/exclusion being defined by each user.

To support the op directory redesign, two new files were added to the top-level of the SIV system directory. ReadmeSiv.txt contains installation and some usage instructions. This file is also included in section 7.1.1.1 of the SIV SOM. Sivuser.mak is a makefile for creating the user working directories. It's usage is described in the ReadmeSiv.txt file.

Improved operability by replacing the two scripts "go.sh" and "cleanup.sh" with "siv.sh". Cleanup is completely automatic regardless if the user exits by signal (<ctrl-c>) or by the command "term abort". This script also handles metric data gathering by saving usage information to a file and if connected to the outside world, sending email to veregge@isds-server.jpl.nasa.gov.

The new support files used by "siv.sh" are "msw.sh", "cleanup_msw.sh", and "ftok". These support files are generic and replace the MSW scripts "go.sh" and "cleanup.sh", "rmshm.sh", and "mskill". The script "msw.sh" calls "cleanup_msw.sh" on termination, so the user need only be familiar with "msw.sh". The additional advantages to theses scripts are that only the IPCs created by this instance of running msw.sh are removed and only the processes created by this instance of running msw.sh are terminated with TERM signals (and it works every time, leaving no orphan processes or IPCs). These scripts also have more complete error checking, signal handling, and recovery than the scripts they replace. The sudo(1) usage has been reduced to one call to improve security and simplify turning the usage of sudo off. When running as root, new files owned by root will have their ownership changed to the user on termination.

The RID file read code was completely rewritten, fixing all known bugs and adding error checking. SIV will no longer accept RID files with errors. This eliminated the use of the file lib/sivrd.c and added the new files datatype.c, datatype.h, library.h, pr_block_def.c, pr_msg.c, pr_stream.c, rid_file.c, rid_file.h, rid_fread.c, str_conv.c, str_conv.h, str_par.c, and str_par.h (in the directories lib or include).

The validation half of dataval was rewritten fixing all known bugs, adding the missing support for using the numeric format specified by the RID files, and adding the missing support for the BCD and Fixed Point data types (all in the function Dataval(), in the file dataval/dataval.c). *However, the data conversion half of dataval is still missing the support for those same two types and only partially supports the signed data type.*

The RID file reading portion of sivmgr was rewritten to use the new RID file reading code in the library (in the function getridinfo() in the file sivmgr/mgrfat.c). No algorithmic changes were made other than those already contained in the RID file library code.

Most know bugs from versions 1.2.5 and 1.3.0 (internal version) have been fixed. The one notable exception being dataval’s partial support for the Signed type and no support for the BCD and Fixed Point types.

<u>Changes to the RID</u> (from section 6.4 in the SOM)

Corrections to the manual are in plain text, <u>while true changes to the RID definition are in underlined text.</u>

### 6.1 - RID File ASCII Format

The RID files have the naming convention “<name>.rid”, where name has a 8 character maximum length, it must follow the normal naming conventions of the disk operating system, and all the letters must be *lower-case*.

Value Representation Formats:

Integer representations follow the standard “C” representations. A hexadecimal value has a “0x” <u>or “0X”</u> prefix. <u>An octal value has a “0” prefix.</u> A decimal value has no prefix. Commas, or any punctuation (except an optional leading “+” or “-” sign in a signed integer), are not permissible in integers (for example 10,000).

Floating point representations follow the IEEE standard, using digits, a decimal point, the letter “e” or “E”, and “+” or “-” sign leading the number or the letter “e”. Commas are not permissible. For example, “11”, “-1.1”, “+.1”, “1.1e2”, “1e-2”, and “0E+2”, are all legitimate -- while “10,000.01” is not.

Text words or strings are white-space delimited sequences of characters. A text word may be a quoted string, which can contain space characters. Quoted strings use the double quote characters: “...”, <u>but the quotes are not word delimiters</u>. The double quote character may be included in a text word by escaping the quote: \”. <u>Spaces may also be escaped, for example, ‘c\ ab’ is equivalent “c ab”.</u> (The single quote character is used here only for illustration and has no special meaning.)

Unless otherwise noted, keywords, field names, data types, actions, and other forms of text entry are not case-sensitive.

#### 6.1.1.1 - Transmission Record Keyword Descriptions

The keywords with their limits and ranges are listed in Table 6-1. Ranges listed as "<integer> =" may have decimal, octal, or hexadecimal record values. In version 1.3.1, version 1.2.5 keyword names are accepted.

##### 6.1.1.1.1 - General Keywords

| | |
|---|---|
| ssend_count | This was *count*. |
| ssend_delay | This was *freq*. |

##### 6.1.1.1.2 - 890-131 & 890-201 Protocols Keywords

| | |
|---|---|
| block_type | The value NON was NONE. |
| proc_code_dest | This was *dst_code*. |
| proc_code_src | This was *src_code*. |
| select_lan | This was *lan_sel*. |
| select_protocol | This was *proto*. |
| sdb_data_type | This was *data_type*. |
| sub_block_id | This was *sub_id*. |

**6.1.2 - Field Record Description**

The general limitations for the field records are: 1 record per line, 255 characters per record, and 500 records per RID file.

The 500 record limit may be extended to 2999 records, by loading (SLOAD) the 500+ record RID by itself and removing (SREM) it before loading any other RIDs. SIV will enforce this restriction of only one 500+ record stream at a time.

General Syntax:

<name> <length>[@<offset>] <type> [<action>] [<value> …]

where:

name — The *name* for the field record is a text word of 2 to 13 characters. The *name* must start with a letter, be unique within the RID file, and cannot be a keyword name.

length — The specification is "[<bytes>:][<bits>]". The maximum data *length* is 32 bytes or 256 bits.

offset — The specification is "[<bytes>:][<bits>]". Raw data segments and following data records are automatically positioned. Whenever an offset is used, a diagnostic or error message will be issued.

For both *length* and *offset*, the value is the *sum* of the byte and bit values.

type — Type limits and ranges are listed in Table 6-2.

action — Action limits and ranges are listed in Table 6-3.

value … The number of values is dependent on the action for the field, but cannot exceed 16.

A *record value* is any value following an action in a RID file or set with the FVAL and FACT directives. A *field value* is the value generated by SIV for that field for transmission to a subsystem. A data type, an action, and the action's *record values* determine the generated *field values*.

An integer record value's format (number base) determines the format in the various SIV outputs, such as the Validation report. If the values are hexadecimal, octal, or decimal, the format of the outputs will be the same.

Values that specify other field records may be field indices, field names, or the letters {P, N, E} (for previous, next, and end field, respectively.

#### 6.1.2.1 - Data Type descriptions

The types are fully described and include bit length limits and value implementation limits. The limits are also in Table 6-2.

*B - Binary Coded Decimal Type*

The range of values is $( -2^{(m \% 4)} \times 10^{(m / 4)}, 2^{(m \% 4)} \times 10^{(m / 4)} )$, where *m* is the field's bit length. The maximum value supported by the SIV is limited by 18 decimal digits. Record values must be decimal. Floating point BCDs are not implemented. Very large values will suffer from least significant bit loss.

*C - Character Type*

String values with lengths greater than the field length will be truncated with a warning. String values with lengths less the field length will be silently left-justified and padded with spaces. The character string will not be NUL (zero) terminated.

*F - IEEE Floating Point Type*

The field length must be 32 bits for single-precision or 64 bits for double-precision values. There is no 48 bit IEEE float.

*F<n> - Fixed Point Scaled Integer Type*

The range of values is [ $-2^{(m-1-n)}$, $2^{(m-1-n)} - 1/2^n$ ], where *m* is the bit length of the field. The smallest fraction is $1/2^n$. The maximum value supported by the SIV is limited by bit $m \leq 32 + n$ bits, where *n* is [ 1, 31 ]. Fractional values that are not exact multiples of $1/2^n$ will be truncated. Very large values will suffer from least significant bit loss. Negative values cannot be represented in fields larger than 32 + *n* bits.

*I - Signed Integer Type*

The range of values is [ $-2^{(m-1)}$, $2^{(m-1)} - 1$ ], where *m* is the bit length of the field. Negative values cannot be represented in fields larger than 32 bits. Record values may be decimal, octal, or hexadecimal.

*M - Modcomp Floating Point Type*

The field length must be 32 bits for single-precision or 48 bits for double-precision values.

*U - Unsigned Integer Type*

The range of values is [ 0, $2^m - 1$ ], where *m* is the bit length of the field. Record values may be decimal, octal, or hexadecimal.

#### 6.1.2.2 - Action Descriptions

Limits (Limitations), potential pitfalls (Warnings), and implementation notes (Notes) are described in the Action Descriptions. Limits include the supported data types which are also in Table 6-3.

#### 6.1.22.1 - General Actions

P ... Pick Action
<u>This action no longer needs to be followed by a number (P<n>) indicating the number of values</u>.

*S ... Sequence Action*
<u>This action no longer needs to be followed by a number (S<n>) indicating the number of values</u>.

#### 6.1.2.2.2 - Data Block Actions

*L ... Data Length Action*

This action does not support the I (integer) or B (BCD) types .

### 6.1.3 - Raw Data Record Description

The raw data file has the naming convention “<name>.raw”, where name has a 8 character maximum length, it must follow the normal naming conventions of the disk operating system, and all the letters must be *lower-case*.

All data and raw records use the same indexing algorithm. Record indices begin at zero and end at the record count minus 1. Each data word in a raw data segment is one record and can be referenced by index or name like a data record.

<u>Raw records are named “RAWDAT:<m>:<nnn>“, where <m> is the number of the rawdat keyword minus 1 and <nnn> is the rawdat word number minus 1 (e.g. the fifth word in the first raw data segment would be named RAWDAT:0:004).</u>

Data records that follow a rawdat keyword are indexed by adding the segment size of the raw data to the index of the data record that preceded the rawdat keyword. The use of offsets with the bit field length (e.g. 16@4270) is unnecessary except for padding data with leading zero bits.